ANALYTIC SERVICES INC.

AR 70 - 1

JAY E. ISRAEL

February 1970

OPTIMAL GROUP STRATEGIES

81

ANSER REPORT

AR 70-1

OPTIMAL GROUP STRATEGIES

JAY E. ISRAEL

February 1970

This document presents results of work sponsored by the Director of Operational Requirements and Development Plans, DCS Research and Development, Headquarters United States Air Force, under Contract F44620-69-C-0014. These results do not necessarily represent Air Force official opinion or policy

Analytic Services Inc. (ANSER)
5613 Leesburg Pike, Falls Church, Virginia 22041

ABSTRACT

This AMSTR Report generalizes the theory of precommit antimissile defense strategies by analyzing the case in which the defense has the ability to assign interceptors to defend groups of targets instead of individual targets only. The defense's objective is to choose the mix of group sizes and the number of interceptors assigned to each group in such a way that target survival is maximized in the event of attack. In this report, the analytic basis is established for a technique to study group defense strategies. Also, a computational procedure is presented for determining optimal attack and defense allocations numerically and for calculating the expected number of targets surviving. Game theory and linear programing are among the analytic techniques used.

TABLE OF CONTENTS

LIST OF FIGURES

TABLE OF CONTENTS—Continued

LIST OF FIGURES—Continued

OPTIMAL GROUP STRATEGIES

I. INTRODUCTION

The scheme the defense uses for deciding at which attacking missiles (penetrators) to direct interceptors is one of the crucial discriminating features of models for analyzing the interaction of ballistic missiles and antimissile missiles (interceptors). In the situation most favorable to the defense, interceptors need not be assigned until the defense has examined the entire attack and determined the intended target of each weapon in the attack. Such target determination requires an impact prediction capability. With a sophisticated command and control system, the defense can then distribute interceptors among the penetrators in such a way that survival of the target system is maximized. If there are more attacking weapons than defending weapons, the best defense strategy is to defend only a subset of the original target system. The subset is chosen by considering the problem parameters. This defense strategy is called "fully responsive."

The offense can thwart a fully responsive defense by dispatching the attacking missiles in waves in such a way that no wave can be seen by the defense until it is too late to intercept previous waves. Denied knowledge of the magnitude of the attack on each target until the attack is almost complete, the defense cannot optimize interceptor allocations against the early attack waves. Consequently, it is unrealistic to assume that the defense has the

capability to implement a fully responsive strategy. Nonetheless, this strategy is frequently studied because it establishes an upper bound on defense efficacy.

Against a wave attack, the defense may choose to oppose every penetrator detected on a first-come-first-served basis. This is called a "subtractive" defense strategy. It requires neither impact prediction capability nor knowledge of the total attack size. Usually, however, it does not achieve as good a result for the defense as can be obtained with other strategies. Suppose, for example, that the number of targets (all identical), attack weapons, and defense weapons are, respectively, T, A, and D, and that

$$A \geq T + D \quad .$$

The offense can then dispatch attacking missiles in two waves: the first, consisting of D weapons, will exhaust the defense; the second, consisting of the remaining weapons, can be allocated uniformly over the now undefended target system. In this way, the defense has no chance to oppose all the penetrators directed at any one target. Furthermore, if there is more than one type of attacking weapon, the offense can save the more effective weapons for the second wave when they will be unopposed.

"Random" defense strategy is one way to improve on a subtractive strategy without the necessity for impact prediction. To implement it, the defense chooses at random which penetrators to oppose from the entire attack. This defense strategy gives a positive probability that all missiles

attacking some of the targets are opposed. Furthermore, if there is more than one type of attack weapon, the defense force will be distributed over all types. In order for random defense to be effective, the defense must know the size of the attack inventory. Overestimating attack size will result in some interceptors' remaining unused after the engagement, while underestimating it may cause the outcome to be more like that of subtractive than that of random defense.

If attack missiles outnumber defense missiles, spreading defense fire power over the entire attack may be of little effect; the defense may be spread too thinly to save any targets.* "Precommit" strategy (also called "preallocate" strategy) is designed to remedy this situation. In it, the defense assigns each interceptor, before hostilities begin, to defend a specific target. The details of the assignment (i.e., how many interceptors are assigned to defend which targets) are kept secret. To be able to use a precommit strategy, the defense must have an impact prediction capability. The computational problem of finding (1) optimal offense and defense weapon assignments and (2) the expected number of targets surviving should both sides use precommit strategies has been solved by J. D. Matheson (Reference 1). (Since that report, he has generalized his results considerably to include heterogeneous mixtures of targets and

*If defense missiles outnumber attack missiles, every penetrator will be opposed if subtractive defense strategy is used, so deviating from subtractive strategy would not benefit the defense.

weapons as well as constraints of interceptor range and attack wave timing.

Precommit attack and defense strategies vary the number of weapons assigned to the different targets. These are the "tapered" strategies often referred to in the literature. (See Reference 1 for more complete details.)

This ANSER Report examines a defense strategy that is more flexible than precommit strategies but follows the same general concept. Specifically, the defender is allowed to partition the targets into groups and to assign a set of interceptors to defend each group. (In addition, there may be one or more undefended groups.) The groups may be of different sizes, and different groups may be defended by different numbers of interceptors. When attack missiles are detected and found to be aimed at any target of a group, they are opposed on a first-come-first-served basis by interceptors assigned to this group. The details of the assignment (i.e., which targets are in which target group and how many interceptors are assigned to defend the group) are kept secret. The objective of the defense is to choose the mix of group sizes and the number of interceptors to assign to each group in order to maximize target survival in case of attack. This is "optimal group" defense strategy.

Interceptor range (or other practical limitations) may impose an upper limit on the number of targets which can be defended as a group. This is one of the constraints of the problem. If the only possible group size is one target, the result is a precommit strategy. An impact prediction

capability of limited accuracy might impose another constraint by setting a lower bound on realizable group sizes. For example, if the defense can discern only that a penetrator is aimed at one of five targets, it is unrealistic to consider groups of fewer than five targets. If the only possible group size is T (i.e., all targets must be in one group), the resultant strategy is subtractive. Consequently, the general optimal group strategy—in which all group sizes are possible—encompasses, among others, precommit and subtractive defense strategies.

Throughout this report, the following assumptions are used:

-1 All targets are identical.

-2 There is only one type of attack weapon and one type of defense weapon.

-3 No more than one interceptor is fired at a single penetrator.

-4 Defense-suppression attacks (e.g., attacks against radars or interceptor bases) are not considered.

-5 Each side knows the characteristics and the quantity of his opponent's weapons.

II. COMPUTATIONAL APPROACH

An attack strategy is denoted by a vector $\vec{A} = (A_0, A_1, A_2, \ldots, A_{\bar{a}})$, which means that A_0 targets are unattacked, A_1 are attacked by exactly one weapon, A_2 by exactly two, etc. The attacker chooses at random which targets to attack and the level of attack against each target. If there are T targets and A attacking weapons, the attack summation conditions

$$\sum_{i=0}^{\bar{a}} A_i = T \quad \text{and} \quad \sum_{i=0}^{\bar{a}} iA_i = A$$

must hold.

In precommit strategies, expected survival is a linear function of each component A_i. Consequently, linear optimization techniques can be employed with the A_i as the variables. However, when target groups larger than one target are considered, expected survival is no longer linear in the A_i (see Reference 2 for an example). One attack linearity remains, however. The attacker in general seeks a mixed strategy

$$\sum_{i=1}^{m} f_i \vec{A}^i ,$$

where each f_i is non-negative and the f_i sum to 1. Each $\vec{A}^i$ is a vector as described above. Since each component in an

$\vec{A}^i$ vector is an integer, each $\vec{A}^i$ is an integer strategy. This is in contrast to a mixed strategy, in which the attacker uses a random device to choose the integer attack strategy for a particular occasion; that is, choosing strategy $\vec{A}^i = (A^i_0, A^i_1, \ldots, A^i_{\bar{a}})$ with probability (or frequency) f_i. Mixed strategies are based on fundamental game theory (see Reference 3, for example). Expected survival is always linear as a function of the f_i.

There are two ways in which the attacker's allocation problem can be treated:

-1 To consider the f_i and A^i_j to be independent variables and use non-linear techniques. If n strategies enter into the final solution, there are $n(\bar{a} + 2)$ variables.

-2 To consider all possible integer strategies $(\vec{A}^i)$—there are a very large number of these. The f_i are the only independent variables and linear techniques can be used. However, the number of variables is astronomical.

The second of these computational approaches has been chosen because algorithms available for linear optimization are more direct and reliable and because the components A^i_j are automatically integers when the second approach is used. Furthermore, experience has made possible a reduction in the number of variables with little degradation in the accuracy of the computation.

A defense strategy can be denoted by a matrix such as

$$\vec{D} = \begin{bmatrix} D_{10} & D_{11} & D_{12} & \cdots & D_{1D} \\ D_{20} & D_{21} & D_{22} & \cdots & D_{2D} \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ D_{T0} & D_{T1} & D_{T2} & \cdots & D_{TD} \end{bmatrix}, \qquad (1)$$

which means that there are D_{10} groups, each group consisting of one undefended target; D_{11} groups, each group consisting of one target and one interceptor; D_{23} groups, each consisting of two targets and three interceptors; and, in general, D_{td} groups, each group consisting of t targets and d interceptors. The symbol D which appears in the subscript of entries in the last column represents the total number of interceptors (defense missiles). As indicated in the matrix, the ranges of the subscripts are $1 \le t \le T$ and $0 \le d \le D$. To avoid ambiguity, all the undefended targets are included in the D_{10} component by prescribing that D_{20}, D_{30}, ..., D_{T0} are equal to zero. When there are constraints on realizable target group sizes, appropriate additional entries of $\vec{D}$ are prescribed to be zero. The defense summation conditions are

$$\sum_{t,d} tD_{td} = T \quad \text{and} \quad \sum_{t,d} dD_{td} = D \ .$$

As will be shown in the next paragraph, the expected number of targets surviving is linear as a function of each component D_{td}. For this reason, linear optimization techniques can be used with the D_{td} as the independent variables, and it is not necessary to treat each integer defense strategy separately. A solution with non-integral values of D_{td} is equivalent to a mixed defense strategy (one in which different integer defense strategies are used with suitable frequencies or probabilities).

The defense problem can be posed mathematically. Suppose $\vec{A}^1, \vec{A}^2, \cdots, \vec{A}^m$ is a list of all possible integer attack strategies. Then let E^i_{td} denote the expected number of targets surviving out of a group of t targets defended by d interceptors if attack strategy $\vec{A}^i$ is used. (A method for computing the constants E^i_{td} will be demonstrated in the next section.) If the attacker uses a mixed strategy such as

$$\sum_{i=1}^{m} f_i \vec{A}^i \quad ,$$

and the defense a strategy such as (1), then the total expected target survival is

$$V = \sum_{i=1}^{m} \left(f_i \sum_{t,d} D_{td} E^i_{td} \right) \quad .$$

The defense's objective is to choose components D_{td} that will maximize V, while the offense chooses frequencies f_i that will minimize it. In this competitive situation, each chooses a strategy that guarantees the greatest expected return for weapons available, regardless of what the opponent does (at least, it is assumed in this report that each side acts in this way and does not take risks to try to capitalize on errors the other side might make). The defense, for example, seeks an optimal strategy $\vec{D}^*$ and a value surviving V^*, such that V^* is the largest number for which

$$\sum_{i=1}^{m}\left(f_i \sum_{t,d} D^*_{td} E^i_{td}\right) \geq V^* \qquad (2)$$

regardless of the attacker's choice of the f_i. The attacker has a corresponding objective: to find optimal frequencies f_i^* and a value surviving, V^{**}, such that V^{**} is the smallest number for which

$$\sum_{i=1}^{m}\left(f^*_i \sum_{t,d} D_{td} E^i_{td}\right) \leq V^{**}$$

regardless of the defender's choice of the D_{td}. A fundamental principle of game theory is that $V^* = V^{**}$ (see Reference 4, pp. 12-25, for example). The defense need not

consider all mixed attack strategies when seeking V^*—rather, only the "pure" strategies, those for which there is only one integer strategy, $\vec{A}^i$ (the frequencies of all others being zero). Indeed, rewriting (2) for each pure attack strategy yields

$$\sum_{t,d} D^*_{td} E^1_{td} \geq V^*$$

$$\sum_{t,d} D^*_{td} E^2_{td} \geq V^*$$

$$\cdot \quad \cdot \quad \cdot \quad \cdot$$

$$\sum_{t,d} D^*_{td} E^m_{td} \geq V^*$$

which implies, for any frequencies f_i summing to 1, that

$$\sum_{i=1}^{m} \left(f_i \sum_{t,d} D^*_{td} E^i_{td} \right) \geq \sum_{i=1}^{m} f_i V^* = V^* \quad .$$

Consequently, if (2) holds for each pure attack strategy, it holds for each mixed attack strategy. When V is transposed in the inequalities, the defense problem is to find

values of D_{td} which maximize V subject to

$$\sum_{t,d} D_{td} E^{i}_{td} - V \geq 0 \qquad (i = 1, 2, \cdots, m)$$

$$\sum_{t,d} tD_{td} = T$$

$$\sum_{t,d} dD_{td} = D$$

$$\text{all } D_{td} \geq 0 \quad .$$

This is a linear programing problem and can be solved by using the simplex method or one of its variants. The attacker's problem is the dual problem of this one. The solution frequencies are a by-product of the defense solution. (They are the coefficients of the "slack" variables in the final defense solution. See Reference 5, pp. 160-164, 275-278, and 479-485, for details.)

III. CALCULATION OF THE SURVIVAL COEFFICIENTS E^i_{td}

Consider one integer attack strategy, $\vec{A} = (A_0, A_1, \cdots, A_{\bar{a}})$. It will yield a series of values E_{td} $(0 < t \leq T;$ $0 \leq d \leq D)$. The superscripts have been dropped because each integer attack strategy is being examined separately—the treatment can be repeated for each one. Then consider what happens to a target group of t targets defended by d interceptors. The configuration of the attack against this group is a matter of chance because, even if the attacker can calculate the optimal mix of defense group sizes, he does not know which targets the defender is assigning to which group. Each possible configuration of the attack on the group can be described by using an integer vector $\vec{a} = (a_0,$ $a_1, \cdots, a_{\bar{a}})$, meaning that exactly a_i targets of the group are attacked by exactly i weapons $(i = 0, 1, 2, \cdots, \bar{a})$.

Clearly, $\sum_{i=0}^{\bar{a}} a_i$ must equal t and each a_i is non-negative and cannot exceed A_i. The probability P that a given attack occurs on the group is given in terms of binomial coefficients by

$$P = \frac{\prod_{i=0}^{\bar{a}} \binom{A_i}{a_i}}{\binom{T}{t}} .$$

Fortunately, there is a recurrence relating these probabilities for different $\vec{a}$ vectors. Specifically, if a_k is replaced

by $a_k + 1$, P is changed to

$$P' = P \frac{A_k - a_k}{a_k + 1} \frac{t + 1}{T - t} \quad . \tag{3}$$

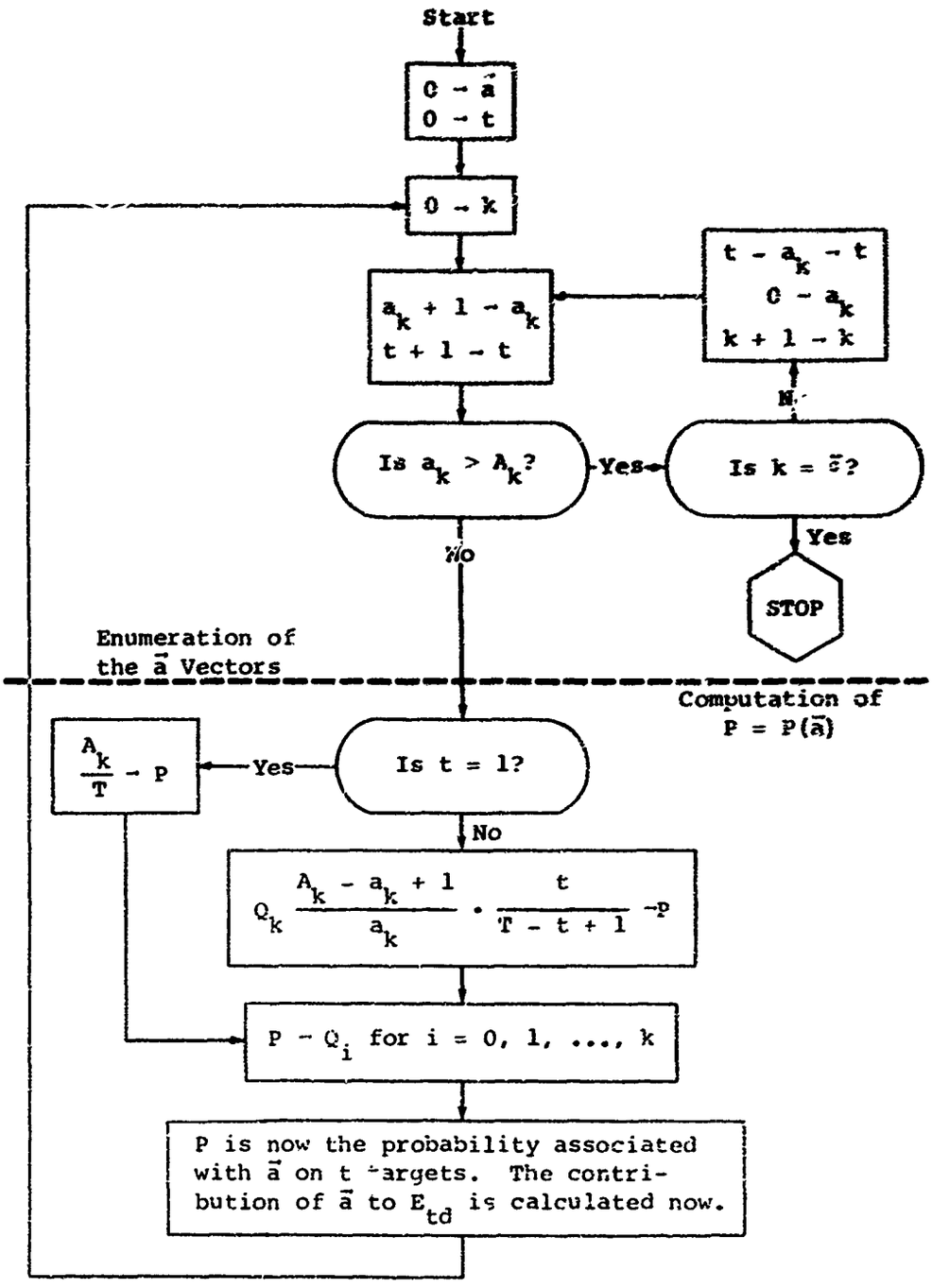
The flow chart in Figure 1 shows how to use this recurrence to calculate the probabilities $P = P(\vec{a})$ for all target group sizes and the many possible configurations of the attack on each. The vectors $\vec{a}$ are enumerated in odometer fashion (the first component a_0 is stepped by one until it can increase no more, at which point the second component increases by one and the first reverts to zero and begins stepping again; the third component is stepped by one when neither the first nor second can increase, at which point the first and second revert to zero; and so on for the fourth and subsequent components). Then, (3) is used to calculate successive P values. The variables $Q_k (k = 0, 1, \cdots, \bar{a})$ in Figure 1 are for temporary storage of values of P needed later.

Now,

$$E_{td} = \sum_{\vec{a}} P(\vec{a}) \; e_{td}(\vec{a}) \tag{4}$$

where $e_{td}(\vec{a})$ is the expected number of survivors from a group of t targets defended by d interceptors in the event that the attack weapons fall on the group in the configuration $\vec{a}$. The value of $e_{td}(\vec{a})$ depends on the order in which attack and defense weapons are fired. For consistency with Reference 2, it is assumed that the attack proceeds in $\bar{a}$

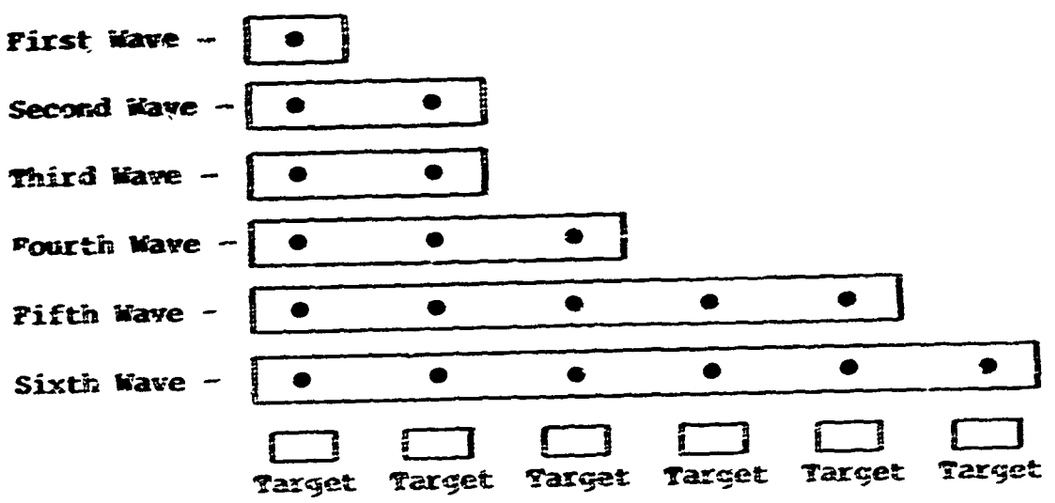
FIGURE 1
CALCULATION OF P = P($\vec{a}$) FOR ALL $\vec{a}$ VECTORS



wzoyo. An example is pictured in Figure 2. It is also assumed that the defense assigns interceptors on a first-come-first-served basis within each target group.

FIGURE 2
ATTACK FIRING DOCTRINE

[In this example, $\bar{A} = (0\ \ 1,\ 2,\ 1,\ 0,\ 1,\ 1)$]



Many alternative attack and defense firing doctrines are possible. Any change alters the constants $e_{td}(\bar{a})$ but leaves the analysis unchanged otherwise.

Calculation of $e_{td}(\bar{a})$ requires s_{ij}, the probability of survival (or expected fraction surviving) of a single target attacked by i weapons, j of which are opposed by interceptors. This report considers the point target case

$$s_{ij} = \left\{1 - P_{KT}(1 - P_I)\right\}^j \left\{1 - P_{KT}\right\}^{i-j} ,$$

where P_{KT} is the single-shot probability of target kill and P_I is the probability of a successful intercept. A target will survive when each unopposed penetrator fails—hence the $\left\{1 - P_{KT}\right\}^{i-j}$ factor—and when each opposed penetrator either fails or is successfully intercepted—hence the $\left\{1 - P_{KT}(1 - P_I)\right\}^j$ factor (0^0 should be interpreted as 1 in this context). A change to area targets or the addition of terminal interceptors changes the constants s_{ij} but leaves the analysis otherwise unaffected.

To derive an expression for $e_{td}(\bar{a})$, suppose that d defense missiles can oppose the first n - 1 waves against the group ($1 \le n \le \bar{n}$) and only δ attack missiles of the n^{th} wave. Which particular attack missiles of the n^{th} wave are

opposed is a matter of chance. If there are $\bar{a}$ attack missiles in the n^{th} wave,

$$e_{td}(\vec{a}) = \sum_{i=0}^{\bar{a}-n} a_i s_{i0} + \sum_{i=\bar{a}-n+1}^{\bar{a}} a_i \left\{ \frac{\delta}{\tau} s_{i,i+n-\bar{a}} + (1 - \frac{\delta}{\tau}) s_{i,i+n-\bar{a}-1} \right\} . \quad (5)$$

The first summation comes from those targets attacked only after the n^{th} wave, so it includes no defense. The second summation reflects the fact that, for each target attacked at a level $i > \bar{a} - n$, the probability is $\frac{\delta}{\tau}$ that $i + n - \bar{a}$ of the weapons attacking it are opposed; otherwise, $i + n - \bar{a} - 1$ are.

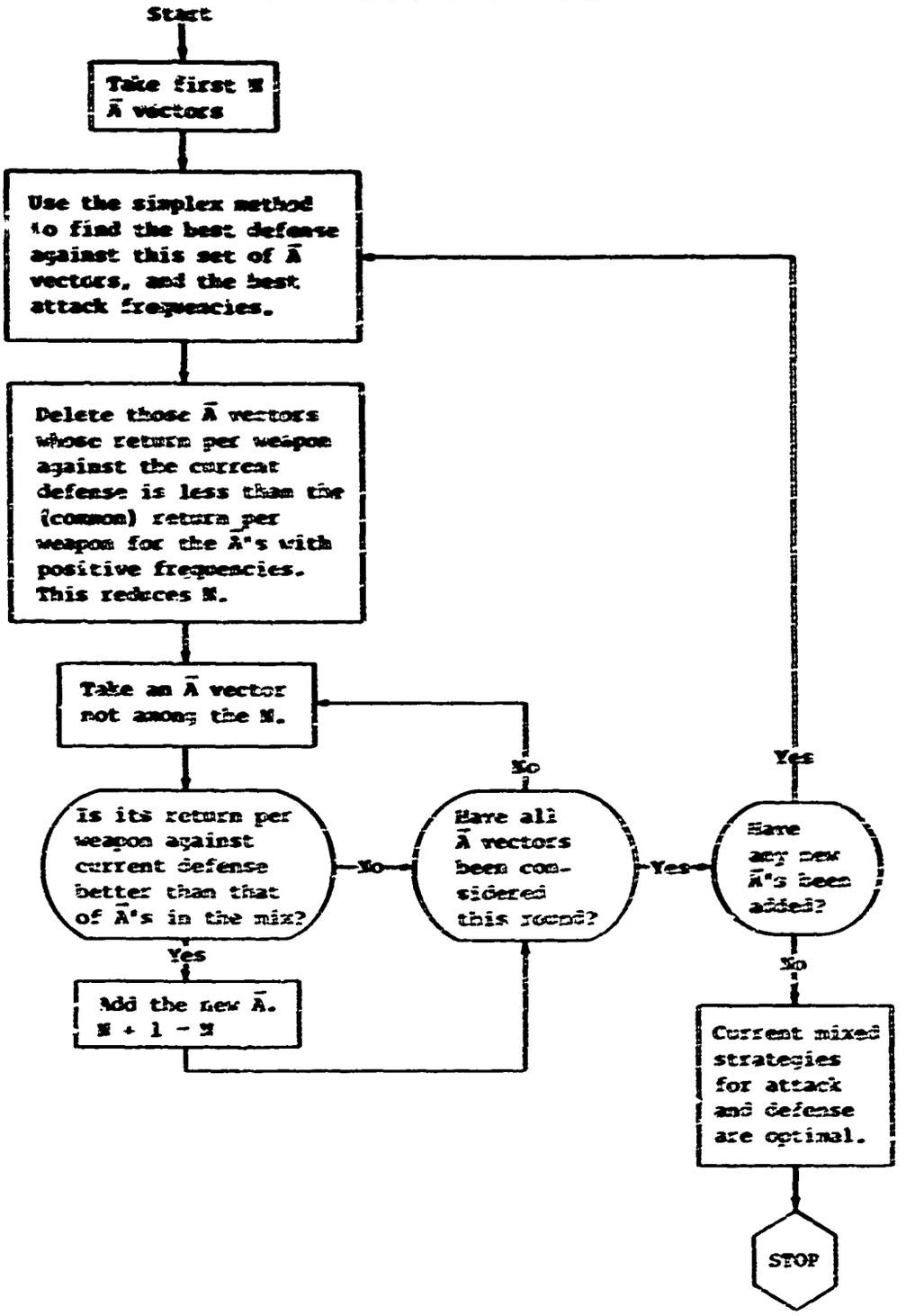
Now that $e_{td}(\vec{a})$ and $P(\vec{a})$ can be calculated, they can be combined using (4) to calculate E_{td} for strategy $\vec{A}$.

IV. COMPUTER PROGRAMS

The first attempt at a computer program for calculating optimal group strategies is called **EXPWAVE**. It is written in PL/I for the Service Bureau Corporation's Call/360 time-sharing system. To enable nontrivial problems to be run within the computer's core limitations, the program does not consider all attack strategies at once. Rather, an iterative approach is used. The computation is first carried out with only a small number N of integer attack strategies. In the solution, only a few of these have non-zero frequencies, and only a few of the components D_{td} are non-zero. The integer attack strategies with non-zero frequencies all yield a common return per weapon to the attack (expected number of targets destroyed per attacking weapon) when used alone against the computed defense strategy. This common value is at least as large as the return per weapon for the integer attack strategies with frequency zero. The computation proceeds by deleting strategies whose return per weapon is less than the common value and by including new strategies which show promise by yielding a higher return per weapon than those already under consideration. The complete process is repeated until no new integer attack strategies show promise, at which point the attack and defense strategies are optimal. The procedure is illustrated in Figure 3.

Despite efforts to conserve core storage, problems with more than a few dozen targets and weapons do not fit in the

FIGURE 3
FLOW CHART OF EXPWAVE



storage space available. This is due primarily to the large number of (t,d) combinations that must be accommodated. The program also requires an unacceptable amount of central processor time to compile and run. Despite these difficulties, EXPWAVE has served two important purposes. First, it has provided a number of correct answers against which subsequent approximate programs can be checked. Second, and more important, it has provided clues as to how to abridge the immense lists of attack and defense strategies with minimal damage to the accuracy of the computation. For example, examination of the output of EXPWAVE reveals that the D_{td} components which emerge as non-zero tend to have d/t about equal to A/T (the attack density) or to D/T (the defense density), whichever is larger. It also reveals that many deleted attack strategies have a return per weapon against the optimal defense practically equal to the final common value (that of the strategies in the optimal mix). This suggests that the attacker could shorten his list of integer strategies without seriously restricting his effectiveness.

Experience with EXPWAVE led to a program called DASH, which considers drastically abbreviated lists of attack and defense strategies and can therefore handle problems with larger numbers of targets and weapons. It does not use the iterative approach of EXPWAVE; it carries out only one set of iterations of the simplex algorithm for each problem.

DASH is written in FORTRAN IV for the General Electric TSS/645 time-sharing system (see Reference 6). A listing

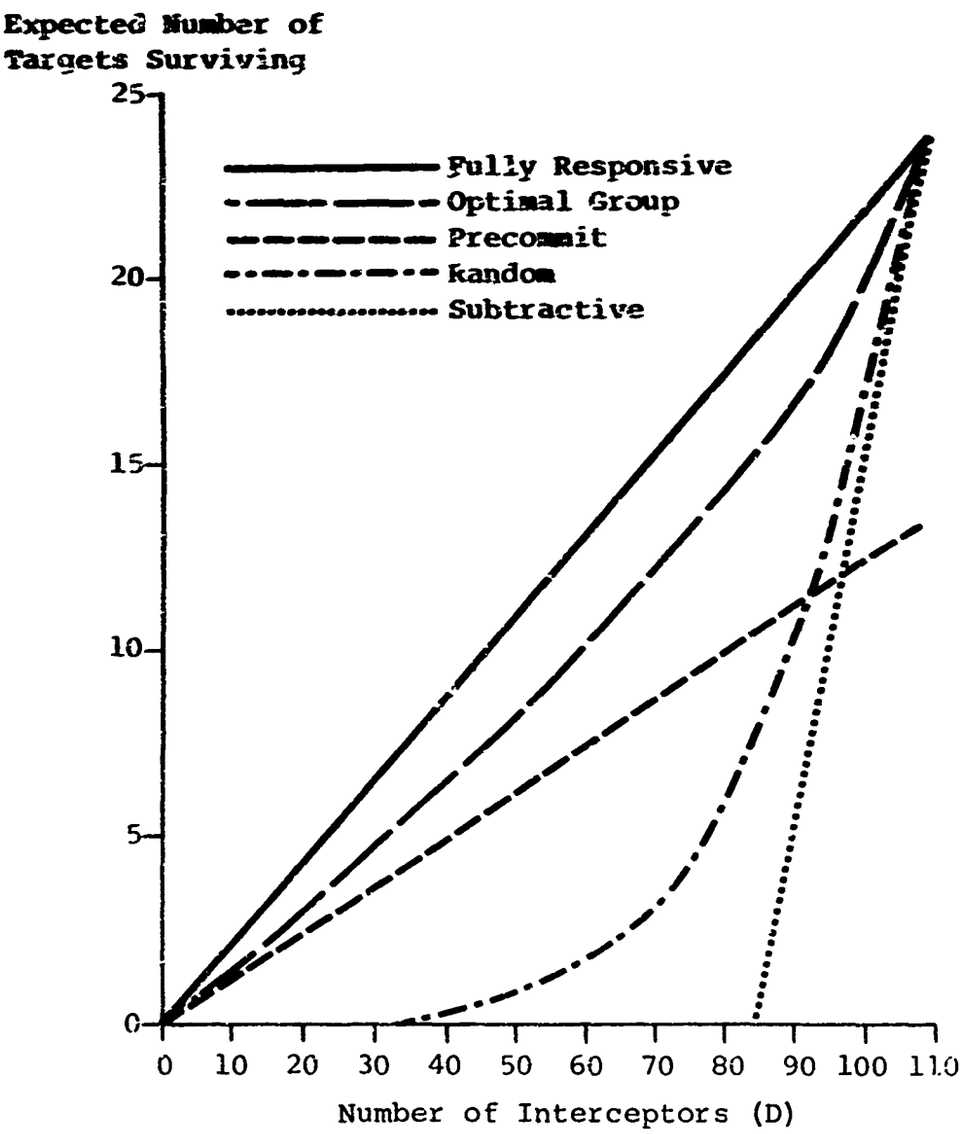
of the program is provided beginning on p. 53. The following table compares typical target-survival results from DASH with the more accurate results calculated by EXPWAVE. DASH results also agree well with special cases solved analytically in Reference 2. Figure 4 compares target-survival results from DASH with those for other defense strategies for a particular case. All data depicted in Figure 4 (except subtractive results, which are easy to compute by hand) were computed by DASH—see Section V, Part A, for details.

ACCURACY OF DASH ON SAMPLE PROBLEMS

T	A	D	P_{KT}	P_I	Expected Number of Targets Surviving		Percent Error in DASH
					EXPWAVE	DASH	
8	16	2	0.75	0.90	1.293	1.293	0.0
8	16	4	.75	.90	2.064	2.082	+ .9
8	16	6	.75	.90	2.772	2.773	+ .1
8	16	8	.75	.90	3.473	3.475	+ .7
8	16	10	.75	.90	4.206	4.206	.0
8	16	12	.75	.90	5.012	5.015	+ .1
5	17	4	.80	.75	.4901	.4901	.0
5	17	6	.80	.75	.7006	.7006	.0
5	17	7	.80	.75	.8543	.8645	+1.3
5	17	8	.80	.75	.9657	.9788	+1.4
5	17	10	.80	.75	1.224	1.225	+0.1
5	17	12	.80	.75	1.486	1.449	-2.6
5	17	14	.80	.75	1.825	1.841	+0.9
5	17	15	.80	.75	1.953	1.967	+0.7
5	17	16	.80	.75	2.077	2.092	+1.4
5	13	12	.80	.75	2.456	2.431	- .7
5	14	12	.80	.75	2.193	2.205	+ .5
5	15	12	.80	.75	1.946	1.954	+ .5
15	20	4	.25	.85	11.092	11.091	- .0
15	20	7	.25	.85	11.676	11.676	.0
15	20	10	.25	.85	12.261	12.260	- .0
15	20	13	.25	.85	12.855	12.855	.0
15	20	16	.25	.85	13.450	13.449	- .0
15	20	18	0.25	0.85	13.850	13.845	-0.0

FIGURE 4
COMPARISON OF DEFENSE STRATEGIES

$T = 24$	$A = 108$	$P_{KT} = P_I = 1$



V. THE DASH PROGRAM

A. Use of the Program*

The DASH program is arranged to function interactively with a user at a teletype. If there is a copy of the program on the systems disc, the program is activated by typing

$FORTRAN MAIN

Otherwise, a paper tape containing the program must be entered first.

The program will print a message at the teletype, requesting the following input values:

T	Number of targets
A	Number of attack missiles
D	Number of defense missiles (interceptors)
AB	(Used for $\bar{a}$) Largest number of attack missiles assignable to a target. The user must use his judgment in choosing this parameter.
MNT	Smallest number of targets permitted in a group
MXT	Largest number of targets permitted in a group
PKT	Probability of target kill by an unopposed penetrator
PI	Probability that an interceptor successfully disables the penetrator against which it is directed
CODE	A nine-digit integer specifying the output desired.

*For a description of the computer system on which DASH is run, see Reference 6. A detailed description of DASH is provided in Appendix A, and a listing of the program begins on p. 53.

Each digit of the nine-digit CODE controls one part of the output and should be either 1—indicating that the corresponding part of the output should be printed—or 0—indicating that the corresponding part of the output should be omitted. When the first digit (starting at the left) is a 1, the output includes a recapitulation of input parameters and the expected number of targets surviving without defense (TSWOD). The latter is often a useful comparison figure—it is calculated with the assumption that the attack is uniform. When the second digit is a 1, the output includes the expected number of targets surviving (TS) when the optimal group strategy is implemented. When the third digit is a 1, expected number of targets destroyed (TD) is printed. The fourth digit corresponds to the defense strategy. When it is 1, the components D^*_{td} [called G(t,d) in the output] are tabulated. When the fifth digit is a 1, the program computes and prints return per weapon to the defense (RWD) for each defense component considered. In the case A > D, this is a check—the RWD values for components in the optimal mix [i.e., components with G(t,d) > 0] should all be equal, and the common value should not be exceeded by the RWD of any other component. If this output option is not chosen, only components with G(t,d) > 0 are printed. If it is chosen, all (t,d) combinations considered are printed and the non-zero ones are marked with an asterisk. If the sixth digit is a 1, the "total attack strategy" (A) is printed; A is the convex combination of integer attack strategies and their frequencies in the optimal solution. Using a 1 as the seventh digit produces component attack

strategies and their frequencies (called F in the output and f_i in this report). If the eighth digit is a 1, the program computes and prints return (against the optimal group strategy) per weapon to the attack (RWA) for each component. This is a check similar to RWD. Using a 1 as the ninth digit produces elapsed time and number of simplex iterations required for the computation.

The second digit of CODE may also be a 3 to indicate that the comparison values TSR (targets surviving with random defense) and TSFR (targets surviving with fully responsive defense) should be printed. Both are calculated under the assumption that the attack is uniform—the best attack strategy in these cases. An approximate result for random defense is calculated by degrading the value of P_{KT}:

$$P'_{KT} = P_{KT}\left[1 - \frac{\min(D,A)}{A} P_I\right] .$$

Fully responsive defense is possible when the defender need not assign interceptors to attack weapons until he sees the entire attack. He should then use interceptors to oppose all the missiles attacking a target for as many targets as possible. This is the basis on which TSFR is calculated.

The following summarizes the digits of CODE.

Position (starting from left)	Output Indicated When the Digit is 1
1	T, A, D, etc.
2	TS (if the digit is 3, TSR and TSFR also)
3	TD
4	G(t,d), called D^*_{td} in Section II
5	ZBD
6	A
7	Component attack strategies and their frequencies
8	BFA
9	Elapsed time, number of simplex iterations

DASH computes precommit strategies if the user specifies MNT = MXT = 1 and subtractive strategies if he specifies MNT = MXT = T. (Why this is so is explained in the introduction to this report.)

For the problem to make sense, the inputs must satisfy

$$0 < PKT,\ PI \leq 1 \quad ,$$

$$MNT \leq MXT \leq T \quad ,$$

and

$$AB > A/T \quad .$$

Current output format and array dimension limitations require

$$AB \leq 20 \quad ,$$

$$0 < T,A,D,MNT,MXT < 10,000 \quad ,$$

and

$$MXT - MNT < \text{about } 110^* \quad .$$

After printing the results of the first problem (or indicating the presence of an error in the input data), the program causes an equal sign to be typed at the left-hand margin to request more data. The user need not enter new values for all input variables; he specifies changes by using the format in the following examples:

```
$CHANGE  D = 25$
$CHANGE  A = 157, D = 73, MXT = 51$  .
$CHANGE  AB = 18, CODE = 1301000111$  .
```

There are also a few special actions the user can take at this point. He can return to the beginning, where values for all input variables must be typed, by typing

```
$CHANGE  CODE = 0$  .
```

He can terminate the running of the program by typing

```
$CHANGE QUIT = 1$  .
```

He can cause the program to become dormant for a desired period of time—for example, 60 seconds—by typing

```
$CHANGE  WAIT = 60$  .
```

---

*This upper limit depends on the number of precommit strategies the program considers, and this depends in a complicated way on the input values.

This input is useful to provide time to think; the computer otherwise disconnects a teletype that remains inactive for a few minutes.

The input and output of a sample run are shown in Figure 5. Inputs have been underlined to distinguish them from outputs.

B. General Structure of the Program

The TSS/645 time-sharing system provides the user with a rather small amount of core storage. Any program of reasonable size must be divided into segments, called links, and loaded into core sequentially by using the CHAIN subroutine. Only one link is in core at any one time. When a new link is activated, it is brought into core, overwriting the link already there. Only the main program and the COMMON data storage area are preserved.

In DASH, the main program (called MAIN—see p. 67) consists of only a few lines. Its job is to call CHAIN. Before returning control to MAIN, each link determines which link will be called next by setting the variable FILE equal to the appropriate link name and FLAG equal to a value which indicates at what point in the link the program is to continue.

There are four links in DASH: files DASH1, DASH2, DASH3, and DASH4. Their responsibilities are as follows:

DASH1—Accepts parameter values; calculates the values s_{ij}; calculates and prints target survival for random and fully responsive defenses.

FIGURE 5
A SAMPLE SESSION WITH THE DASH PROGRAM

```
$FORTRAN MAIN


>----

 T,A,D,AA,MNT,MXT,PKT,PI,CODE=4,8,4,5,1,4,1,1,131111111

PROGRAM: DASH.  VERSION: WA.  TODAY IS: 01/07/70

TSR=     1.000      TSFR=     2.000

   5 ITERATIONS.  ELAPSED TIME:    15.00 SECONDS.

(T,A,D)=(   4.    8,    4) AA= 5 MNT=    1 MXT=    4 PKT=1.00000 PI=1.00000
CODE=   131111111

TSDD=    0.

TS=     1.5000

TD=     2.5000

   T     D    G(T,D)    RHO
   1     0     1.500
   1     1     1.000  0.37500  *
   1     2     0.     0.37500
   1     3     0.     0.31250
   1     4     0.     0.23437
   2     4     0.750  0.37500  *
   3     4     0.     0.23437
   4     4     0.     0.

   F     RHA   LEVEL=0     1     2     3     4     5
0.      0.31250          0     0     4     0
0.      0.28125          0     2     0     2
0.      0.31250          0     2     1     0     1
0.2500  0.31250          0     3     0     0     0     1
0.7500  0.31250          0     1     2     1
```

FIGURE 5—Continued
A SAMPLE SESSION WITH THE DASH PROGRAM

```
LEVEL    A
  0     0.
  1     1.500
  2     1.500
  3     0.750
  4     0.
  5     0.250

>---

=$CHANGE PKT=.25, PI=.5, CODE=130100101$

TSR=     2.641      TSPR=     2.454

   5 ITERATIONS.  ELAPSED TIME:    17.00 SECONDS.

(T,A,D)=(    4,    8,    4) M=  5 CNT=    1 MXT=    4 PKT=0.25000 PI=0.50000
CODE=    130100101

TSMOD=     2.2500

TS=     2.4542

   T     D    G(T,D)
   1     0     1.867
   1     1     0.267
   2     4     0.933

   F  LEVEL=0     1     2     3     4     5
0.4668        0     0     4     0
0.5332        0     1     2     1

>---
```

FIGURE 5—Continued
A SAMPLE SESSION WITH THE DASH PROGRAM

```
=$CHANGE MXT=15

TS=     2.441      TSFR=     2.455

   7 ITERATIONS.  ELAPSED TIME:    15.00 SECONDS.

(I,J,D)=(   4.    8.    4) 43= 5 MNT=    1 MXT=    1 PRT=0.25000 PI=0.50000
SUMS=    130100101

TSMD=     2.2500

TS=     2.4486

   I    D   G(I,D)
   1    0    1.514
   2    1    0.973
   3    2    1.514

   F  LEVEL=0     1     2     3     4     5
0.5676         0     0     4     0
0.4324         0     1     2     1


>---


=$CHANGE MXT=5, D=3$

INPUT ERROR:
MXT>T.
FIX IT.


>---
```

FIGURE 5—Continued
A SAMPLE SESSION WITH THE DASH PROGRAM

```
=$CHANGE MXT=4$

TSR=     3.063      TSFR=     2.484

   2 ITERATIONS.  ELAPSED TIME:    11.00 SECONDS.

G(T,A,B)=(    4,    8,    8) AB= 5 MNT=    1 MXT=    4 PKT=0.25000 PI=0.50000
CODE=    130100101

TSAB=     2.2500

TS=     3.0625

   T     B    G(T,B)
   1     0     0.
   4     8     1.000

   P  LEVEL=0     1     2     3     4     5
1.0000        0     0     4     0


>---


=$CHANGE WAIT=20$

          15
          10
           5
           0


>---


=$CHANGE CODE=0$


>---
```

FIGURE 5—Continued
A SAMPLE SESSION WITH THE DASH PROGRAM

```
T,A,D,AB,MNT,MXT,PKT,PI,CODE=10,100,50,20,1,10,,40,,60,130100101

PROGRAM: DASH.  VERSION: HA.  TODAY IS: 01/07/70

TSR=     0.374      TSFR=     0.905

   5 ITERATIONS.  ELAPSED TIME:  255.00 SECONDS.

(T,A,D)=(  10,  100,   50) AB=20 MNT=    1 MXT=   10 PKT=0.40000 PI=0.60000
CODE=    130100101

TSWOD=     0.0605

TS=     0.8914

   T     D    G(T,D)
   1     0     5.000
   5    50     1.000

   F  LEVEL=0     1     2     3     4     5     6     7     8     9    10
1.0000         0     0     0     0     0     0     0     0     3     4
                     3     0     0     0     0     0     0     0     0     0


>---

=$CHANGE QUIT=1$

PROGRAM STOP AT 630

READY
```

DASH2—Generates the $\bar{A}$ vectors; implements the procedure shown in Figure 1 and the summations of (5) to complete the array of values, E^{i}_{td}.

DASH3—Part 1: initializes the simplex algorithm; chooses the initial basic feasible solution.

Part 2: prints final results.

DASH4—Carries out simplex iterations.

DASH1 can stand alone and calculate survival for random and fully responsive defense strategies. To activate it, the user types

$FORTRAN DASH1

Only the $CHANGE ...$ input format is used in this case, and the only relevant variables are T, A, D, PKT, and PI. (The user must not change the value of CODE.)

C. How to Change the Program

1. The Dimensions of Arrays

The sizes of arrays are indicated by the values of three variables set at the beginning of DASH1—MXR, MXC, and MXAB. MXAB is the largest value of AB (that is, $\bar{a}$) permitted. MXR is two more than the number of integer attack strategies the program considers. MXC is two more than the number of defense components the program permits. (Increasing MXC increases the difference MXT - MNT that the program can handle.) Comment lines near the beginning of each link show how the dimensions of each array depend on these three quantities. (Note, however, that arrays appearing in more

than one link have a comment only in the first link in which they appear.) To change the sizes of arrays, it is necessary to change only the declarations at the beginning of each link and the values of MXR, MXC, and MXAB in lines 70-190 of DASH1.

If MXAB is increased, additional numbers* are required in the DATA statements at the beginning of DASH2. The numbers required are:

$$RT(I) = \sqrt[3]{10I}$$

$$ST(I) = \log_e(I + 1) \quad ,$$

for I = 1, 2, 3, $\cdots$, $(\bar{a} + 1)$. In addition, the data on line 143 of DASH2 should be changed as directed by line 146. Finally, the constants in the subscripts in DASH2, lines 1893-1899, should be changed to multiples of the new MXAB.

To accommodate larger arrays, core may need to be conserved elsewhere, in which case use of the NOLN compilation option (Reference 6, p. 8) should be considered.

2. The "Damage Function"

To change s_{ij} to reflect a different damage function, the user must first alter the input and output statements

---

*These numbers are used in a hash-coding procedure to guarantee that no integer attack strategy is included more than once. The hash-coding technique is used here to assign an arbitrary but unique real number to each vector $\bar{A}$ for the purpose of identification. The values already in the DATA statement came from Reference 7.

to include the parameters on which the new function depends. There must also be room in COMMON to store those parameters. Any new COMMON statements should precede all existing COMMON statements and must be identical in every link and in MAIN. The COMMON statements in DASH1, lines 550-555, must not be overlooked.

The values of s_{ij} are computed in DASH1, lines 180-270, and stored in vector S according to the following scheme:

$$S[(i + 2)(i - 1)/2 + j + 1] = s_{ij}$$

$$1 \leq i \leq \bar{a} \qquad (6)$$

$$0 \leq j \leq i \quad .$$

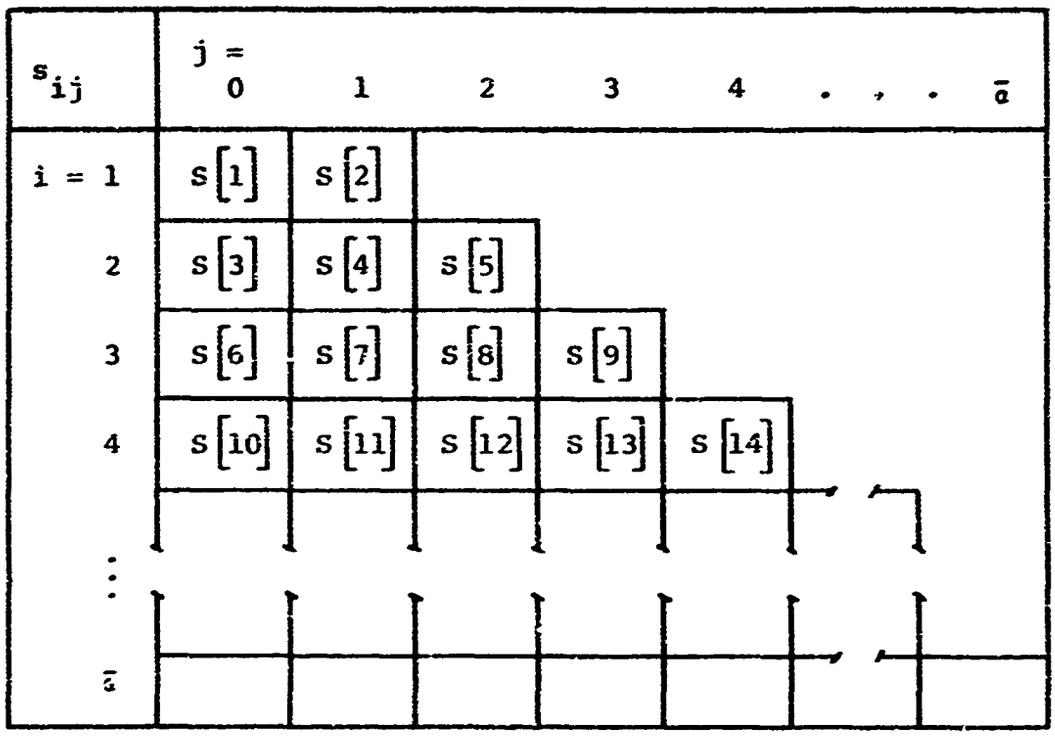
Figure 6 illustrates this scheme in tabular form. To change the damage function, then, one deletes lines 180-270 and substitutes statements to calculate s_{ij} ($1 \leq i \leq AB$; $0 \leq j \leq i$), storing the results in S as just indicated.

3. The "Firing Doctrine"

The quantities $e_{td}(\bar{a})$ are calculated in DASH2, lines 590-765. The calculation can be changed by altering this portion of the program. When this portion is initiated, the vector $\bar{a}$ is stored in the array A [a_I = A(I + 1)]; d is stored in D, t in T, and $\sum_{i=0}^{\bar{a}} ia_i$ (total number of attack weapons on a specific target group) in AT. NU is the position in the array A of the last non-zero entry. It

may be assumed that MU is greater than one. Values of s_{ij} are stored in the array S as indicated in (6) and Figure 6. Any set of instructions replacing 590-765 must calculate $e_{td}(\vec{a})$ and store the result in the variable called SUC in the program.

FIGURE 6
RELATION BETWEEN s_{ij} AND S



APPENDIX A

DETAILED DESCRIPTION OF THE BASE PROGRAM

APPENDIX A

DETAILED DESCRIPTION OF THE CODE PROGRAM

1. Introduction

Following are internal program equivalents of some of the variables mentioned in this report

Variable	T	A	D	$\bar{c}$	CCOE	$\bar{A}$	$\bar{a}$	Q
Internal representation	ST	SA	SD	SB	LEVEL	W	A	TEMP

The coefficient matrix of the linear programing problem is calculated by the program and stored in the array Z, which can be represented as

$$Z = \begin{bmatrix} 1 & t_1 & \cdots & t_n & T & c \\ 0 & d_1 & \cdots & d_n & D & 0 \\ p_{1,0}^{1} & p_{t_1,d_1}^{1} & \cdots & p_{t_n,d_n}^{1} & 0 & -1 \\ \vdots & \vdots & & \vdots & \vdots & \vdots \\ p_{1,0}^{m} & p_{t_1,d_1}^{m} & \cdots & p_{t_n,d_n}^{m} & 0 & -1 \end{bmatrix} .$$

Each column, except the last two, corresponds to a (t,d) combination. The last column contains the coefficients of the variable V (expected number of targets surviving). The

next to last column is an artificial variable of the form (t,d) = (T,D) and is used as the starting point (initial basic feasible solution) of the simplex method. The entries of this column are so contrived by the program that the defense will not employ the component corresponding to the column (except in degenerate cases in which all targets are destroyed regardless of defense strategy).

Each row of Z corresponds to a constraint of the linear programing problem. The first two rows are the summation conditions

$$\sum_{t,d} dD_{td} = T \quad \text{and} \quad \sum_{t,d} dD_{td} = D \quad .$$

Each additional row corresponds to an integer attack strategy and contains the values of Σ^{i}_{td} in the appropriate columns.

2. Description of Program Listing

Line numbers refer to the listing beginning on p. 52.

Line Numbers	BASIC
70-90	Initialize array sizes.
110-170	Accept and check parameters typed by user. XS is the number of "precommit" strategies (t = MNT) used.
180-270	Calculate s_{ij} and store in S.
271-297	Calculate and print random and fully responsive results.

Line Numbers

293-479	Decide on (t,d) combinations and fill first two rows of Z.
480-500	Initialize appropriate variables. Set FLAG equal to 1 and go to BASE2.
505-690	Accept input which is in the format $CHANGE ...$.
700-790	Check inputs and print error messages.
806-840	Calculate $\Sigma(1 - P)^J$ for random and fully responsive defenses. This subroutine calculates, for various values of I, J, and P, the expected number of targets surviving from I targets, each attacked by J missiles which each have probability P of destroying a target.

BASE2

152-158	Activate an abnormal return if error is detected.
160-220	Fill last column of Z. Normal return invokes BASE3.
230-300	Implement procedure shown in Figure 1 for the attack strategy corresponding with row number NR of the Z matrix. Because of possible underflow, the array TEMP (that is, Q) has been augmented by another array, ITEMP. Entries in TEMP remain between 0.1 and 1, and entries in ITEMP reveal by what power of 0.1 TEMP must be multiplied to obtain the proper value.

Line Numbers

520-800	Calculate $e_{ad}(\vec{a})$ (called ZEC here) by Formula (5), multiply by $P(\vec{a})$, and add into the proper variable, E_{ad} [c.f. Equation (4)].
820-1040	Generate integer attack strategies.
880-1030	Generate for each integer i ($A/T \leq i \leq E$) a strategy with the i^{th} component (a_i) as large as possible (provided every target is attacked at least once, if possible). Remaining attack weapons are distributed uniformly among remaining targets. These are "rectangular" attacks.
1040-1860	Generate strategies with different general shapes. First, some of the attack missiles are allocated in an attack vector of the form (0, ..., 0, i, ..., i, j, i, ..., i). Then, remaining attack missiles are added judiciously to reflect the influence of weighting functions (stored in the array WT) to obtain a variety of shapes. IM is the internal representation of i. KAP is the position of the j entry in the attack vector. -MD is the number of i entries on either side of j. LAM is the number of attack missiles added. ID is the identifier of the weighting function, which for $1 \leq ID \leq 8$ is stored in WT. ID = 9 means always add attack missiles to the targets

Line Numbers

	attacked least. ID = 10 causes attack missiles to be added in a pseudo-random way.
1870-1890	Generate uniform attack strategy. This is always the first one (corresponding to the third row of Z).
1891-1899	Calculate and store weighting functions.
1900-1915	Initialize appropriate variables.
1920-1990	Use hash coding to avoid repeated integer attack strategies. The code for each strategy is stored (temporarily) in the last column of Z.
2000-2030	Return to earlier part of DASE2 with a new strategy or, if called from outside DASE2, return to the location indicated in PEFI and PEFL.

DASE3

The linear programing problem is solved using the revised simplex method, as described in Reference 5, pp. 507-519. The matrix called B^{-1} in that reference is stored in array B of DASE3 and DASE4. There is a slack variable for each integer attack strategy, but the slack variables are not included in Z. Instead, the program allows for them in its logic.

BASE identifies the basis variables by number (i.e., column number of Z). Initially, these are the slack variables, the artificial variable (t,d) = (T,D), and variable number one

[(t,d) = (1,0)]. The first iteration introduces V into the basis and deletes one of the slack variables. IND(I) is set equal to 1 when the I^{th} variable is in the basis, and 0 otherwise.

Line Numbers

120	Defines a subroutine to dissect LEVEL (CODE).
140-156	Initialize appropriate variables.
160-190	Print diagnostic output. See Section 3 of this appendix.
200-220	Alter entries of Z to conform with the fact that DASH4 operates on the idea of "minimize value destroyed" instead of on "maximize value surviving."
250-430	Initialize appropriate variables.
440-490	Decide which slack variable is replaced by V.
490-660	Initialize certain variables for simplex algorithm.
670-690	Start simplex procedure.
730-770	Print preliminary information.
720-830	Store defense solution in IND, attack solution in U.
850-940	Print results.
950-970	Calculate probability of survival of an undefended target using the optimal attack strategy.
980-1120	Print optimal defense strategy, and calculate and print return per weapon to the defense for each defense component.

Line Numbers

1130-1260	Initialize appropriate variables.
1300-1340	Call DASH2 to reconstruct integer attack strategies.
1343-1360	Activate an abnormal return if error is detected.
1365-1380	Compute the convex combination of the integer attack strategies. Store temporarily in the next to last column of Z.
1390-1505	Print frequency, EMA, and integer attack strategy.
1510-1520	Print the convex combination of integer attack strategies. At this point, the program has looped through the whole list of attack vectors used.

DASH4

100-210	Update essential variables and arrays for next simplex iteration.
330-480	Choose incoming basis variable.
490	Test to determine whether optimum has been found.
500-680	Choose outgoing basis variable. If none can be found, error is indicated.
690-750	Prepare for next iteration.
760-778	Try a smaller tolerance, EPS, in hope that error was due to roundoff.
780-800	Return to DASH3, line 730.

3. Diagnostic Output

The user may append an extra digit to CODE, to the left of those described in Section V. Presence of that digit is a signal to the program to print certain intermediate values generated during the course of the computation. If that digit is 1 or higher, the final printing of integer attack strategies includes the values of MU, KAP, LAM, ID, and IM (see DASH2, lines 1040-1860) associated with each. If it is 2 or higher, the entire Z matrix is printed, showing the values of E^i_{td}. If it is 3, a detailed trace is provided of the process of generating the integer attack strategies. If this last option is chosen, the user must press the "return" key every few lines when an equal sign is typed out. If the extra digit is absent from CODE, the program assumes it to be zero and the user may omit the extra digit unless he desires diagnostic output.

3. Diagnostic Output

The user may append an extra digit to CODE, to the left of those described in Section V. Presence of that digit is a signal to the program to print certain intermediate values generated during the course of the computation. If that digit is 1 or higher, the final printing of integer attack strategies includes the values of MF, KAP, LAM, ID, and IN (see DASH2, lines 1040-1860) associated with each. If it is 2 or higher, the entire Z matrix is printed, showing the values of E^i_{tz}. If it is 3, a detailed trace is provided of the process of generating the integer attack strategies. If this last option is chosen, the user must press the "return" key every few lines when an equal sign is typed out. If the extra digit is absent from CODE, the program assumes it to be zero and the user may omit the extra digit unless he desires diagnostic output.

LISTING OF DASH PROGRAM

SOURCE DASH1

```
* JAY ISRAEL.  PROGRAM: DASH. REVISED 4 FEB. 70.
COMMON FILE,FLAG,BT,BA,BD,AB,MNT,MXT,PKT,PI,NR,NC,Z(28,120),LEVEL,
& NWFL,PWFL,MXR,MXC,MXAB,TIM,XS,MU,KAP,LAM,ID,IX,KID,KR
COMMON LA,SUC,V(21),S(230),P1,RAT,I,J,L,I1,I2,X,ASC(2)
ASCII ASC
FILENAME FILE,NWFL
INTEGER FLAG,BT,BA,BD,AB,V,PWFL,XS
IF(FLAG.NE.1.AND.FLAG.NE.2)LEVEL=20000000
IF(FLAG.NE.1)GO TO 9992
MXR=28
MXC=120
MXAB=20
* DIMENSIONS Z(MXR,MXC),V(MXAB+1),S(MXAB*(MXAB+3)/2)
1 PRINT 10
10 FORMAT("->----"/"- T,A,D,AB,MNT,MXT,PKT,PI,CODE")
READ: BT,BA,BD,AB,MNT,MXT,PKT,PI,LEVEL
FLAG=836371
6 MU=MNT
CALL TEST(MINO(BT,BA,BD,AB,MNT,LEVEL).GE.0.AND.AMIN1(PKT,PI).GT.0.
& ,"INPUTS SHOULD BE >0.",5,$9992)
KR=MINO(BD,AB*MNT)
IF(BA.LT.BT)MU=(BA*MNT)/(2*BT)-1
MU=MINO(MU,KR)
CALL TEST(AB.LE.MXAB,"AB TOO BIG.",3,$9992)
CALL TEST(MXT.LE.BT,"MXT>T.",2,$9992)
CALL TEST(MNT.LE.MXT,"MNT>MXT.",2,$9992)
LAM=MINO(KR,MU+MAXO(MNT*BA/BT+1,MXC-MXT+MNT-4),MU+(AB-1)*MNT)
XS=LAM-MU+1
CALL TEST(PKT.LE.1..AND.PI.LE.1.,"PKT OR PI >1.",4,$9992)
CALL TEST(AB.GE.BA/BT+1,"AB TOO SMALL.",4,$9992)
NC=MXT-MNT+XS+1
CALL TEST(NC.LE.MXC-2,"(MXT-MNT) TOO BIG.",5,$9995)
CALL DATE-TIM(ASC,TIM)
IF(FLAG.EQ.836371)PRINT 11,ASC
11 FORMAT(" PROGRAM: DASH.  VERSION: 2A.  TODAY IS: ",2A4)
K=1.
P1=1.-PKT*(1.-PI)
RAT=(1.-PKT)/P1
```

```
DO 2 I=1,NA
 X=X*PI
 L=I*(I+3)/2
 S(L)=X
 DO 2 J=1,I
 S(L-1)=RAT*S(L)
2 L=L-1
IF((X0)(LE/EL/1000000.10).LE.1)GO TO 19
20 LA=NA/NT+1
LAX=LA+1
I2=NA-NT*LAX
I1=NT-I2
NR=MIN0(NA,NT)
X=PKT*(1.-FLOAT(LA)*PI/FLOAT(NA))
RAT=SUA(I1,LAX,X)+SUA(I2,LA,X)
X=PKT*(1.-PI)
SUC=RAT
IF(LAX.LE.0.OR.NR.GE.NA) GO TO 18
J=NR/LAX
IF(J.GE.I1)GO TO 9
I=NR-J*LAX
SUC=SUA(J,LAX,X)+SUA(I1-J-1,LAX,PKT)+SUA(I2,LA,PKT)
& +SUA(1,I,X)+SUA(1,LAX-I,PKT)
GO TO 18
9 J=(NR-I1*LAX)/LA
I=NR-J*LA-I1*LAX
SUC=SUA(I1,LAX,X)+SUA(J,LA,X)+SUA(I2-J-1,LA,PKT)
& +SUA(1,I,X)+SUA(1,LA-I,PKT)
18 PRINT 12,RAT,SUC
12 FORMAT(" TSR=",F9.3,5X," TSFR=",F9.3)
IF((LE/EL.EQ.2)1000000)GO TO 9992
19 I=MIN0(NT,NA)
I1=NC+1
I2=NC+2
Z(1,I1)=I
Z(2,I1)=NR
Z(1,I2)=0.
Z(2,I2)=0.
 DO 5 J=3,MXN
 Z(J,I1)=I
 DO 5 L=1,NC
 5 Z(J,L)=0.
Z(1,1)=1.
Z(2,1)=0.
DO 8 I=1,XS
Z(1,I+1)=ANT
Z(2,I+1)=MU
8 MU=MU+1
```

```
      IF(MXT.EQ.MXT)GO TO 7
      J=XS+2
      I1=XAT+1
      I2=MAX0(30,-20)
      DO 4 I=J,NC
      Z(1,I)=I1
      Z(2,I)=AMIN1(FLOAT(3),AMAX1(AINT(.5+FLOAT(I2*I1)/FLOAT(3I)),1.))
    4 I1=I1+1
    7 FLAG=1
      FILE="DASH0"
      NR=2
      NI=AB/2+1
      NU=0
      RETURN
 9995 PRINT:"EXCESS=",NC-MXC+2
 9992 CALL NEXT
      IF(LEVEL.NE.2000000) IF(LEVEL)6,1,6
      CALL TEST(MIN0(MT,NA.30+1).GT.0.AND.AMIN1(PKT,P1,1.-PKT,1.-PI).GE.
     & 0.,"  ",1,$9992)
      GO TO 20
      END
      SUBROUTINE NEXT
      COMMON FILE,QUIT,T,A,D,AB,MNT,MXT,PKT,PI,NR,NC,Z(28,120),CODE,
     & PHFI,WAIT
      FILENAME FILE,PHFI
      INTEGER T,A,D,AB,CODE,QUIT,WAIT
      NAMELIST/CHANGE/T,A,D,AB,MNT,MXT,PKT,PI,CODE,QUIT,WAIT
    2 PRINT 100
  100 FORMAT("->----"/"-")
      QUIT=0
      WAIT=0
      READ CHANGE
      IF(QUIT.NE.0)STOP
      IF(WAIT.EQ.0)RETURN
      DO 1 I=5,WAIT,5
      CALL ZSLEEP(5)
    1 PRINT:"",WAIT-I
      GO TO 2
      END
      SUBROUTINE TEST(C,M,L,*)
      LOGICAL C
      ASCII M(L)
      IF(C)RETURN
      PRINT 1,M
      PRINT 2
      RETURN 1
    1 FORMAT("0INPUT ERROR:"/1H ,12A4)
    2 FORMAT(" FIX IT.")
      END
```

```
FUNCTION SUM(I,J,P)
SUM=I
IF(J.GT.0.AND.I.GT.0)SUM=SUM*(I.-P)**J
RETURN
END
*  END OF DASH1

[illegible] DASH1,(SORT)
FTNTRAN =DASH1,(CERAIX.NOSE)
[illegible] DASH2
COMMON FILE,FLAG,ST,[illegible],AB,[illegible],[illegible],PKT,PI,NE,NC,Z(28,120),LEVEL,
& [illegible],PHFL,[illegible],[illegible],[illegible],[illegible],X5,MU,KAP,LGM,ID,IN,MID,KR
COMMON LA,SUC,V(21),S(230),T,AT,TEMP(21),A(21),NU,I,J,K,ASP1,W,
& TAU,P,LAC1,D,[illegible],S1,S2,S3,T1,W1,W2,L,ITEMP(21),ZT(160)
FILENAME FILE,PHFI
INTEGER FLAG,ST,[illegible],[illegible],AB,TAU,W,V,A,T,AT,PHFL,D,[illegible],W1,W2,ASP1,XS,ZT
LOGICAL TRACE,END
DIMENSION RT(21),ST(21),NN(8)
* DIMENSIONS (RT,ST,TEMP,ITEMP)(MZAB+1),ZT(8*MXAB)
NAMELIST/DUMP/I,[illegible],XS,[illegible],KAP,LAM,ID,IN,MID,KR,LA,SUC,V,A,T,AT,
& ITEMP,TEMP,[illegible],J,K,[illegible],TAU,P,D,[illegible],[illegible],S1,S2,S3,T1,W1,W2,L
DATA RT/2.154425,2.714418,3.107233,3.419952,3.684031,3.914868,
& 4.121285,4.305769,4.481405,4.641589,4.791420,4.932424,5.065797,
& 5.192494,5.313293,5.428835,5.539658,5.645216,5.748897,5.848035,
& 5.943922/
DATA ST/0.69315,1.09861,1.38629,1.609441,1.79176,1.94591,2.07944,
&  2.19722,2.30259,2.39790,2.48491,2.56495,2.63906,2.70805,
&  2.77259,2.83321,2.89037,2.94444,2.99573,3.04452,3.09104/
DATA NN/1,21,41,61,81,101,121,141/
* NN(I)=MXAB*(I-1)+1
END=ZA.LE.MAXO(ZD,ST)
GO TO 9992
9992 PRINT DUMP
FILE="DASH1"
FLAG=2
RETURN
70 FILE="DASH3"
J=NC+2
DO 71 I=3,NR
71 Z(I,J)=-1.
FLAG=1
RETURN
```

```
00220 3 IF(SUC.EQ.0.)GO TO 70
00230 73 NR=NT+1
00240 Z(NR,1)=FLOAT(V(1))/FLOAT(ST)
00250 AT=0
00260 T=0
00270 J=NA3+1
00280 DO 4 I=1,J
00290 4 A(I)=0
00300 5001 K=1
00310 5 IF(W(K).EQ.0) IF(K-LA)6,6,9992
00320 A(K)=A(K)+1
00330 T=T+1
00340 AT=AT+K-5
00350 IF(T.NE.1) GO TO 8
00360  W5=K
00370  P=FLOAT(V(K))/FLOAT(ST)
00380  I=(K-2)*(K+1)/2+1
00390  IF(K.GT.1) Z(NR,I)=Z(NR,I)+P*S(I)
00395  J=0
00400  GO TO 14
00410 8 IF(A(K).LE.W(K).AND.I.LE.NT) GO TO 9
00420  IF(K.GE.LAP1)GO TO 9992
00430  AT=AT-(K-1)*A(K)
00440  T=T-A(K)
00450  A(K)=0
00460 6 K=K+1
00470  GO TO 5
00480 9 P=TEMP(K)*FLOAT(T*(W(K)-A(K)+1)/FLOAT(A(K)*(ST-T+1))
00481 J=ITEMP(K)
00482 14 IF(P.GE..1)GO TO 13
00483 J=J+1
00484 P=10.*P
00485 GO TO 14
00486 13 IF(P.LE.1.)GO TO 7
00487 J=J-1
00488 P=.1*P
00489 GO TO 13
00490 7 DO 11 I=1,K
00495 ITEMP(I)=J
00500 11 TEMP(I)=P
00510 *
00520 IF(I.LT.NT.OR.J.GT.20)GO TO 5001
00525 IF(J.NE.0)P=P*.1**J
00530 M=T-NT+2
00540 IF(I.GT.NT)M=M+XS-1
00550 106 IF(NI.GT.1)GO TO 100
00560 Z(NR,M)=Z(NR,M)+P+FLOAT(T)
00570 GO TO 105
```

```
00580 100 M=MIN(AT,IFIX(Z(2,M)+.1))
00590 S1=A(1)
00600 T1=T-A(1)
00610 W2=AT-T1
00620 L=1
00630 W1=AT
00640 DO 101 N=2,MM
00650  IF(D.GT.W2)GO TO 102
00660  S1=S1+A(N)*S(L)
00670  L=L+1
00680  W1=W2
00690  T1=T1-A(N)
00700 101 W2=W2-T1
00710 102 S2=0.
00720 S3=0.
00730 DO 103 I=N,W1
00740 S2=S2+A(I)*(S(L+1)-S(L))
00750 S3=S3+A(I)*S(L)
00760 103 L=L+1
00765 SUC=S1+FLOAT(D-W2)*S2/FLOAT(T1)+S3
00770 IF(SUC.GT.1.E-30) Z(NR,M)=Z(NR,M)+P*SUC
00780 105 M=M+1
00790 IF(M.LE.MC.AND.IFIX(Z(1,M)+.1).EQ.T)GO TO 106
00800 GO TO 5021
00810 *
00820 9999 LA=-122349402
00830 9992 SUC=0.
00835 IF(NR.EQ.NRR.AND.MU.NE.0)GO TO 201
00840 ASP1=NXA2+1
00850 DO 202 L=1,ASP1
00860 202 V(L)=0
00865 TRACE=LEVEL.EQ.3000000000.AND.FLAG.EQ.1
00870 IF(MU)2?3,250,270
00880 270 LA=MU
00890 MU=MU+1
00895 IF(LA.LE.1)GO TO 271
00900 IF(LA.GT.AB)GO TO 208
00910 I=(BA-BT)/(LA-1)
00920 IF(I.GT.0)GO TO 203
00930 271  I=BA/LA
00940 IF(I.LE.0) GO TO 208
00950 203 TAU=BT-I
00960 B=BA-I*LA
00970 IF(TAU.LT.0.OR.(TAU.EQ.0.AND.A.GT.0)) GO TO 270
```

```
01930 J=0
01290 IF(TAU.LT.0) J=W/TAU
01300 V(J+2)=X-J*TAU
01310 V(J+1)=TAU-V(J+2)
01320 V(LA+1)=I
01330 GO TO 220
01340 293 IF(KAP.GT.1)IF(LA+1523494402)290,2291,290
01343 MU=0
01345 GO TO 290
01350 2291 IF(IN.LE.0)GO TO 290
01352 IF(S3.EQ.SUC)IN=0
01354 IF(SUC.GT.0.)S3=SUC
01355 IN=IN/2
01356 GO TO 291
01360 293 KAP=3A/3T+2
01370 215 MU=0
01380 KAP=KAP-1
01390 IF(KAP.EQ.1)MU=1
01400 LAM=3A-(KAP-1)*3T
01420 209 MU=MU+1
01430 ID=0
01440 IF(KAP+MU.LE.0.OR.KAP-MU.GT.AB.OR.3T+2*MU+1.LT.0)
01450 $  IF(KAP-1)201,201,215
01455 IF(TRACE>PRINT:"+ KAP,J,M,MU",KAP,LAM,MU
01460 290 ID=ID+1
01465  S3=-1.
01470 IF((ID.GT.1.AND.LAM.EQ.0).OR.ID.GT.10)GO TO 209
01475 IF(TRACE>PRINT:"ID=",ID
01480 IN=FLOAT(3T)/FLOAT(1-2*MU)+.001
01490 291 J=3T+2*IN+MU
01500 IF(J.LT.0)GO TO 2291
01505 IF(TRACE)PRINT:"IN,J=",IN,J
01506 IF (TRACE)READ:I
01510 V(KAP)=J
01540 IF(MU.EQ.0.OR.IN.EQ.0)GO TO 219
01550 LA=-MU
01560 DO 210 J=1,LA
01570 TAU=KAP-J
01580 V(TAU)=IN
01590 TAU=KAP+J
01600 210 V(TAU)=IN.
01610 219 MU=MIN0(MU,-1)
01620 LA=KAP-MU-1
01630 IF(V(1).LT.LAM.OR.(KKO.AND.LAM.GT.0)) GO TO 221
01640 V(1)=V(1)-LAM
01650 V(2)=V(2)+LAM
01660 GO TO 220
```

```
01370 221 XL=LAM-V(1)
01375 IF(SKO)MU=LAM
01380 DO 211 K=1,MU
01385 200 LAP1=MAX0(LA+1,4)
01400 K=1
01410 IF(ID.GT.8)GO TO 92
01420 V=V(ID)
01430 J=MT(K)*V(1)
01440 92 DO 212 I=2,LAP1
01450 IF(V(I).LE.0) GO TO 212
01460 IF(ID-9)95,99,30
01470 99 K=1
01480 GO TO 206
01490 90 K=1
01500 P=RINT+ST(KKP)-ST(LAP1+MU+1)
01505 IF(TRACE.PRINT) P=-P
01510 IF(P-AINT(P))-.5)212,212,206
01520 95 K=K+1
01530 L=V(1)-MT(K)
01540 IF(L.LE.J)GO TO 212
01550 K=1
01560 J=L
01570 212 CONTINUE
01720 206  IF(K.EQ.4.AND.K.LE.2.OR.V(K).LE.2)GO TO 213
01740 V(K+2)=V(K+2)+1
01750 V(K)=V(K)-2
01760 V(K-1)=V(K-1)+1
01770 LA=MAX0(LA,K+1)
01780 GO TO 211
01790 213 V(K+1)=V(K+1)+1
01800 V(K)=V(K)-1
01805 LA=MAX0(LA,K)
01810 IF(.NOT.(KO.AND.K.EQ.1)) GO TO 200
01830 211 CONTINUE
01835 IF(SKO) GO TO 220
01840 V(2)=V(2)+V(1)
01850 V(1)=0
01860 GO TO 220
01870 250 LA=4A/3T+1
01880 V(LA+1)=3A-3T*(LA-1)
01890 V(LA)=3T-V(LA+1)
```

```
01891 DO 91 I=1,45
01892 ST(I)=I
01893 WT(I+20)=I
01894 WT(I+40)=ARR(I)
01895 WT(I+60)=(I-MID)*(I-MID)+1
01896 WT(I+80)= I*WT(I+40)
01897 WT(I+100)=I+I
01898 WT(I+120)= WT(I+40)*WT(I+40)
01899 91 WT(I+140)=500/I
01900 KR=LA
01910 KR=2
01915 IF(FLAG.EQ.1)CALL SETERR(31,1,$9000)
01920 222 LAP1=LA+1
01930 DO 232 I=1,LAP1
01940 232 IF(V(I).GT.0)SUC=SUC+RT(I)*ANCD(FLOAT(V(I)),ST(I))
01945 IF(ST.ACE)PRINT 299,SUC,(V(I),I=1,LAP1)
01950 IF(KR.EQ.2)GO TO 231
01960 DO 232 I=3,KR
01970 232 IF(SUC.EQ.Z(I,MXC)) IF(KU)9990,9992,9992
01980 231 KR=KR+1
01990 Z(KR,MXC)=SUC
02000 201 IF(FLAG.EQ.1)GO TO 3
02010 FILE=PHFI
02020 FLAG=PHFL
02030 RETURN
02035 299 FORMAT(" SUC=",1PE14.7,"  V=",(12I4))
02040 END
02050 * END OF DASH2

SAVE DASH2.(SORT)
FORTRAN =DASH2.(CHAIN,NOGO)
SETERR DASH3
00010 COMMON FILE,FLAG,BT,BA,BD,AB,MNT,MXT,PKT,PI,NR,NC,Z(28,120),LEVEL,
00020 * PHFI,PHFL,MXR,MXC,MXAB,TIM,XS,KU,KAP,LAM,ID,IN,MID,KR
00030 COMMON LA,SUC,V(21),BASE(28),B(28,28),PF,NCP1,NCP2,IC,EPS,IND(145)
00040 * ,U(28),R,K,PBT,PBD,LIM,Q1,Q2,ASC(2)
00050 INTEGER FLAG,BT,BA,BD,PHFL,V,BASE,PF,R,AB,XS
00060 * DIMENSIONS BASE(MXR),U(MXR),B(MXR,MXR),IND(MXC+MXR-2)
00070 FILENAME PHFI,FILE
00080 REAL IN0
00090 LOGICAL CHOICE,Q1,Q2,Q3
00100 ASCII ASC,NIL
00110 DATA NIL/00/
00120 CHOICE(I,J)=MOD(LEVEL/I,10).EQ.J
00130 GO TO (9991,9992,9993,9994),FLAG
```

```
00140 9991 FST=ST
00150 F=0=30
00153 NCP1=NC+1
00156 NCP2=NCP1+1
00160  IF(LEVEL.LT.2000000000)60 T0 100
00170  PRINT 68
00180  D0 110 I=1,NR
00190  110 PRINT 69,,Z(I,J),J=1,NCP2)
00200 100 D0 101 I=3,NR
00210  D2 101 J=1,NC
00220  101 Z(I,J)=Z(1,J)-Z(I,J)
00250 IC=1
00260 EPS=1.E-5*FBT
00270 LIM=NC+NR
00280 D0 102 I=2,NC
00290 102 IND(I)=0.
00300 IND(1)=1.
00310 IND(NCP1)=1.
00340 D0 103 I=3,NR
00350 103 U(I)=-1.
00360 D2 104 I=NCP1,LIM
00370 104 IND(I)=1.
00380 IND(NCP2)=0.
00390 J=NR-2
00400 D0 105 I=1,J
00410 105 BASE(I)=NCP2+I
00420 BASE(NR-1)=NCP1
00430 BASE(NR)=1
00440 R=3
00450 IF(NR.E0.3) G0 T0 109
00460 D0 106 I=4,NR
00470 IF(Z(I,1).GT.Z(R,1))R=I
00480 106 C0NTINUE
00490 109 X=Z(2,NCP1)
00500 Y=Z(1,NCP1)
00510 D0 107 I=1,NR
00520 D0 108 J=3,NR
00530 K=I+2
00540 B(I,J)=0.
00550 IF(K.E0.J) B(I,J)=1.
00560 108 C0NTINUE
00570 IF(I.GE.NR-1) G0 T0 107
00580 B(I,2)=(Z(K,1)*Y-Z(K,NCP1))/X
00590 B(I,1)=-Z(K,1)
00595 U(I)=-1.
00600 107 C0NTINUE
00603 U(NR)=0.
00606 U(NR-1)=0.
```

(DASH3 Continues)

```
00610 B(NR-1,2)=1./X
00620 B(NR,2)=-Y/X
00630 B(NR-1,1)=0.
00640 B(NR,1)=1.
00645 R=R+2
00650 PF=R
00660 K=NCP2
00670 FILE="DASH4"
00680 FLAG=1
00690 RETURN
00700 *
00710 9993 PRINT:"ERROR IN SIMPLEX. "
00720 *
00730 9992 IF(CHOICE(1,0)) 60 TO 200
00740 CALL DATE*TIM(ASC,X)
00750 PRINT 50,IC,2600.*(X-TIM)
00755 IF(EPS.NE.1.E-5*FBT)PRINT:"EPS=",EPS
00760 200 IF(CHOICE(100000000,1))PRINT 51,BT,BA,BD,AB,MNT,MXT,PKT,PI,
00770 & LEVEL,BT*(1.-Z(3,1))
00780 DO 201 J=1,LIM
00790 201 IND(J)=0.
00800 DO 202 I=1,NR
00810 J=BASE(I)
00820 IND(J)=FBT*B(I,1)+FBD*B(I,2)
00830 202 U(I)=-B(PF,I)
00850 Q1=CHOICE(100000,1).OR.IND(NCP1).NE.0.
00860 Q2=CHOICE(10000,1)
00870 IF(.NOT.CHOICE(10000000,0))PRINT 52,BT-IND(NCP2)
00880 IF(CHOICE(1000000,1))PRINT 53,IND(NCP2)
00890 IF(.NOT.(Q1.OR.Q2)) GO TO 210
00900 PRINT 54
00910 Q3=.NOT.Q2
00920 IF(Q1)PRINT 55
00930 IF(Q2)PRINT 56
00940 IF(Q3) GO TO 231
00950 TIM=0.
```

(DASH3 Continues)

```
00960 DO 230 I=3,NR
00970 230 TIM=TIM+U(I)*(Z(I,1)-Z(I,1))
00980 231 IF(Q1) PRINT 70,IND(1)
00990 DO 203 J=2,NCP1
01000 IF(IND(J).EQ.0..AND.(Q3.OR.J.EQ.NCP1)) GO TO 203
01010 I=Z(1,J)+.1
01020 IC=Z(2,J)+.1
01030 PRINT 57,I,IC
01040 IF(Q1) PRINT 58,IND(J)
01050 IF(Q3) GO TO 203
01060 X=0.
01070 Y=Z(1,J)
01080 DO 204 I=3,NR
01090 204 X=X+U(I)*(Y-Z(I,J))
01100 X=(X-Y*TIM)/Z(2,J)
01110 PRINT 59,X
01115 IF(IND(J).GT.0)PRINT 64
01120 203 CONTINUE
01130 210 Q1=CHOICE(100,1)
01140 Q2=CHOICE(10,1)
01150 IF(.NOT.(Q1.OR.Q2.OR.CHOICE(1000,1)))GO TO 250
01160 ASC(1)=NIL
01170 IF(Q2)ASC(1)=4H
01180 ASC(2)=ASC(1)
01190 PRINT 60
01200 IF(Q2)PRINT 61
01210 R=MIND(A2,10)
01220  IF(Q1) PRINT 62,(I,I=1,R)
01230 PF=3
01240 MU=0
01250 DO 211 I=1,MXR
01260 211 Z(I,NCP1)=0.
01300 224 FLAG=2
01310 FILE="DASH2"
01320 PHFL=4
01330 PHFI="DASH3"
01340 RETURN
01343 219 PRINT:"V'S DON'T MATCH."
01346 GO TO 250
01350 9994 K=LA+1
01360 IF(SUC.EQ.0.)GO TO 219
01365 Q3=U(PF).GE.1.E-4
01370 DO 213 I=1,K
01380 213 IF(V(I).GT.0)Z(I,NCP1)=Z(I,NCP1)+FLOAT(V(I))*U(PF)
01390 IF(Q2.OR.(Q1.AND.Q3))PRINT 63,U(PF)
01400 IF(.NOT.Q2)GO TO 214
01410 X=0.
```

(DASH3 Ends)

```
01420 DO 215 J=1,NC
01430 215 X=X+Z(PF,J)*IND(J)
01435 X=X/FLOAT(BA)
01440 PRINT 59,X
01450 214 IF(.NOT.(01.AND.(02.0R.93))) GO TO 223
01460 R=MIND(K,11)
01470 PRINT 65,(V(I),I=1,R)
01480 R=(K-2)/10
01490 IF(R.GT.0) PRINT 66,(ASC,(V(10*I+J),J=2,11),I=1,R)
01495 IF(LEVEL.GE.1000000000)PRINT 49,MU,KAP,LAM,ID,IN
01500 223 PF=PF+1
01505 IF(PF.LE.NR)GO TO 224
01510 220 IF(CHOICE(1000,1))PRINT 67,(I,Z(I+1,NCP1),I=0,AB)
01520 250 FILE="DASH1"
01530 FLAG=2
01540 RETURN
01550 *
01560 68 FORMAT("0MATRIX=")
01570 69 FORMAT(/(8F9.3))
01580 70 FORMAT("    1    0",F9.3)
01585 49 FORMAT(30X,"MU=",I2," KAP=",I2," LAM=",I4," ID=",I2," IN=",I4)
01590 50 FORMAT(1H0,I4," ITERATIONS.  ELAPSED TIME:",F8.2," SECONDS.")
01600 51 FORMAT("0(I,A,D)=(",2(I4,","),I4,") AB=",I2," MNT=",I4,
01610 & " MXT=",I4," PKT=",F7.5," PI=",F7.5/" CODE=",I12/"OTSWOD=",
01612 & F10.4)
01620 52 FORMAT("OTS=",F10.4)
01630 53 FORMAT("OTD=",F10.4)
01640 54 FORMAT("0   I    D")
01650 55 FORMAT("&   G(I,D)")
01660 56 FORMAT("&   RWD")
01670 57 FORMAT(2(1X,I4))
01680 58 FORMAT("&",F9.3)
01690 59 FORMAT("&",F8.5)
01700 60 FORMAT("0   F  ")
01710 61 FORMAT("&  RWA   ")
01720 62 FORMAT("&LEVEL=0",10I5)
01730 63 FORMAT(1X,F6.4)
01740 64 FORMAT("& *")
01750 65 FORMAT("&",2X,11I5)
01760 66 FORMAT(7X,2A4,7X,10I5)
01770 67 FORMAT("0LEVEL   A"/(I4,F9.3))
01780 END
01790 * END OF DASH3

$SAVE DASH3.(SORT)
$FORTRAN =DASH3,(CHAIN,NOGO)
```

(DASH4 Begins)

```
$ENTER DASH4
00010 COMMON FILE,FLAG,BT,BA,BD,AB,MNT,MXT,PKT,PI,NR,NC,Z(28,120),LEVEL,
00020 & PHFI,PHFL,MXR,MXC,MXAB,TIM,XS,MU,KAP,LAM,ID,IN,MID,KR
00030 COMMON LA,SUC,V(21),BASE(28),B(28,28),PF,NCP1,NCP2,IC,EPS,IND(146)
00040 & , U(28),R,K,FBT,FBD,LIM,I,J,X,Y
00050 INTEGER FLAG,BT,BA,BD,PHFL,V,BASE,PF,R,AB,XS
00060 + DIMENSIONS BASE(MXR),U(MXR),B(MXR,MXR),IND(MXC+MXR-2)
00070 FILENAME PHFI,FILE
00080 REAL IND
00100 400 I=BASE(R)
00110 IND(I)=0.
00120 IND(K)=1.
00130 BASE(R)=K
00140 DO 401 J=1,NR
00150  DO 402 I=1,NR
00160   IF(I.EQ.R)GO TO 402
00190  X=B(I,J)+U(I)*B(R,J)
00193  IF(ABS(X).LT.1.E-4) X=0.
00196  B(I,J)=X
00200  402 CONTINUE
00210 401 B(R,J)=B(R,J)/U(R)
00330 403 K=0
00340 X=-EPS
00350 DO 410 J=1,LIM
00360  IF(IND(J).NE.0.) GO TO 410
00370  Y=0.
00380  IF(J.LE.NCP2) GO TO 417
00390  I=J-NC
00400  Y=-B(PF,I)
00410  GO TO 416
00420 417 DO 418 I=1,NR
00430 418 Y=Y-B(PF,I)*Z(I,J)
00440 IF(J.EQ.NCP2) Y=Y+1.
00450 416 IF(Y.GE.X) GO TO 410
00460  K=J
00470  X=Y
00480 410 CONTINUE
00490 IF(K.EQ.0) GO TO 4001
00500 IF(K.LE.NCP2) GO TO 425
00510 L=K-NC
00520 DO 420 I=1,NR
00530 420 U(I)=B(I,L)
00540 GO TO 426
00550 425 DO 422 I=1,NR
00560  U(I)=0.
00570  DO 422 J=1,NR
00580  422 U(I)=U(I)+B(I,J)*Z(J,K)
00590 426 X=1.E30
```

(DASH4 Ends, MAIN Begins)

```
00595 W=0.
00600 R=0
00610 DO 430 J=1,NR
00615  I=J
00620 IF(U(I).GE.EPS.AND.PF.NE.I) GO TO 431
00623 W=AMAX1(W,U(I))
00626 GO TO 430
00630 431 Y=(FRT*B(I,1)+FBD*B(I,2))/U(I)
00640  IF(Y.GT.X)GO TO 430
00650  X=Y
00660  R=I
00670 430 CONTINUE
00680 IF(R.EQ.0) GO TO 4000
00690 DO 435 I=1,NR
00700  IF(I.EQ.R) GO TO 435
00710  IF(ABS(U(I)).LT.EPS) U(I)=0.
00720  U(I)=-U(I)/U(R)
00730 435 CONTINUE
00740 IC=IC+1
00750 GO TO 400
00760 4000 IF(W.EQ.0)GO TO 4003
00765 EPS=.5**3
00770 GO TO 426
00777 4003 FLAG=3
00778 GO TO 4002
00780 4001 FLAG=2
00790 4002 FILE="DASH3"
00800 RETURN
00810 END
00820 * END OF DASH4

$SAVE DASH4,(SORT)
$FORTRAN =DASH4,(CHAIN,NOGO)
$ENTER MAIN
00010 COMMON FILE,FLAG,MISC(440R)
00020 FILENAME FILE
00030 INTEGER FLAG
00050 FILE="DASH1"
00060 FLAG=1
00070 1 CALL CHAIN(FILE)
00080 GO TO 1
00090 END
00100 * END OF MAIN PROGRAM

$SAVE MAIN,(SORT)
$FORTRAN MAIN,(NOLN)
```

APPENDIX B

PROGRAM EXTENSION FOR IMPERFECT INFORMATION

APPENDIX B

PROGRAM EXTENSION FOR IMPERFECT INFORMATION

An extended version of the DASH program analyzes the effects on results when strategy is based on erroneous information about the opponent's weapons. Called "Version WAII" (Wave Attack—Imperfect Information), the new program is useful for studying the sensitivity of expected target survival to errors in intelligence estimates.

Version WAII operates in a way similar to that of the original version (Version WA, called "DASH" in the body of this report) except that there are three sets of parameter values—one set for each of three phases. The value of T (number of targets) is common to all three phases.

The first phase operates with the values of the parameters as the attacker thinks they are and produces attack and defense strategies based on these values exactly as in Version WA. The second phase operates with the parameter values as the defender thinks they are and produces attack and defense strategies based on these values. The third phase employs the attack strategy from the first phase against the defense strategy from the second phase to determine the resulting expected target survival.

The input parameters for Phase I corresponding to A, D, AB, MNT, MXT, PKT, PI, and CODE of Version WA are A1, D1, AB1, MNT1, MXT1, PKT1, PI1, and CODE1. Similarly, Phase II uses input parameters A2, D2, AB2, MNT2, MXT2, PKT2, PI2, and CODE2.

It is assumed that the attacker knows A and AB precisely and that the defender knows D, MXT, and MXT precisely. However, neither the defender nor the attacker need be assumed to have perfect information about the effectiveness of his own or his opponent's weapons; new input parameters, PKT3 and PI3, allow values of PKT and PI that can be different from the values of PKT1, PKT2, PI1, and PI2. So, the parameters the program uses in Phase III are A1, D2, AB1, MXT2, MXT2, PKT3, PI3, and CODE3.

Since MXT and MXT are among the parameters about which the attacker may be uncertain, the new program can answer questions such as: "What happens if the defender adopts an optimal group strategy but the attacker thinks that the defender is using a preallocate strategy?"

Version WAII performs the function of Version WA when the user sets CODE2 and CODE3 equal to zero. Then, the program skips Phases II and III and the user works with T, A1, D1, AB1, MXT1, MXT1, PKT1, PI1, and CODE1 only.

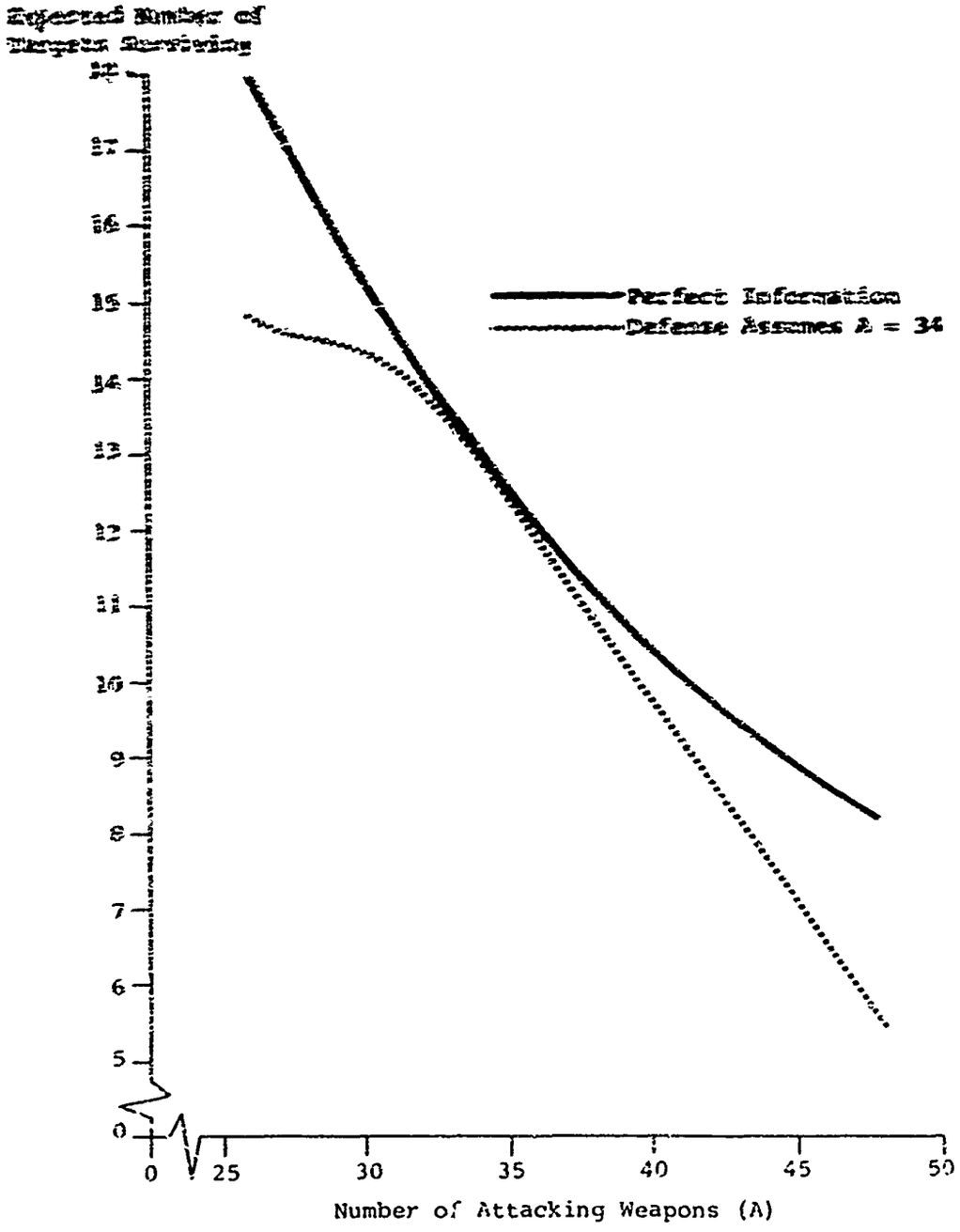
Figure 7 illustrates one way in which the new program can be applied. The objective in this example is to determine how far from optimal the expected target survival is when the defense estimates the number of attacking weapons incorrectly. The upper curve is a plot of target survival as a function of attack size, which can be obtained by using Version WA. The lower curve, produced with the use of Version WAII, is calculated for a defense assumption that there are 34 attacking weapons. The difference between the

FIGURE 7

CONSEQUENCE OF IMPERFECT INFORMATION: AN EXAMPLE

T = 20 P_{kT} = 0.90 MRT = 1

D = 25 P_T = 0.95 MRT = 20



curves indicates, as a function of attack size, the penalty to the defense for estimating attack size incorrectly.

A listing of Version WAIT is not included in this report. The new program differs from the one described in the body of this report mainly in the first link, where input values are manipulated. Figure 8 shows the input and output of a sample run of Version WAIT of DASH. Inputs have been underlined to distinguish them from outputs. Interested parties may contact Analytic Services Inc., Strategic Branch, for further details.

FIGURE 8

A SAMPLE SESSION WITH WAII VERSION OF DASH PROGRAM

```
$FORTRAN MAIN

>---

T,CODE1,CODE2,CODE3=4,11110010I,11110010I,11110010I

A1,B1,AB1,MNT1,MXT1=12,8,10,1,4

A2,B2,AB2,MNT2,MXT2=15,10,11,2,4

PKT1,PI1=.75,.9

PKT2,PI2=.865,.888

PKT3,PI3=.41,.56901

PROGRAM: DASH.  VERSION: WAII.  TODAY IS 02/10/70

...PHASE 1

  15 ITERATIONS.  ELAPSED TIME:    11.00 SECONDS.

(T,A,D)=(   4,   12,    8) AB=10 MNT=    1 MXT=    4 PKT=0.75000 PI=0.90000
CODE=    111100101

TSWOD(UA)=     0.0625      TSR=     1.3720      TSFR=     1.8124

TS=     1.7227

TD=     2.2773

   T     D    G(T,D)
   1     0     0.869
   1     1     0.187
   1     2     0.439
   1     3     0.591
   2     6     0.088
   3     8     0.579
```

FIGURE 8—Continued

A SAMPLE SESSION WITH MAIN VERSION OF DASH PROGRAM

```
   F  LEVEL=0    1    2    3    4    5    6    7    8    9   10
0.1208       0    2    0    0    0    2
0.1594       0    0    3    0    0    0    1
0.1856       0    0    2    1    0    1
0.0692       0    1    0    1    2
0.4550       0    1    0    2    0    1

...PHASE 2

  16 ITERATIONS.  ELAPSED TIME:    15.00 SECONDS.

(T,A,D)=(   4,   15,   10) AB=11 MNT=   2 MXT=   4 PKT=0.86500 PI=0.88800
CODE=   111100101

TSWOD(UA)=     0.0035      TSR=     0.7969      TSFR=     1.5016

TS=     1.2777

TD=     2.7223

   T    D   G(T,D)
   1    0    1.399
   2    4    0.225
   2    6    0.049
   2    7    0.485
   2   10    0.541

   F  LEVEL=0    1    2    3    4    5    6    7    8    9   10
0.2274       0    0    1    2    0    0    0    1
0.3431       0    0    3    0    0    0    0    0    0    1
0.1502       0    1    2    0    0    0    0    0    0    0    1
0.2793       0    0    0    2    1    1

...PHASE 3

   0 ITERATIONS.  ELAPSED TIME:     8.00 SECONDS.

(T,A,D)=(   4,   12,   10) AB=10 MNT=   2 MXT=   4 PKT=0.41000 PI=0.56901
CODE=   111100101

TSWOD(UA)=     0.8215      TSR=     1.9306      TSFR=     1.9607

TS=     1.8077

TD=     2.1923
```

FIGURE 8—Continued
A SAMPLE SESSION WITH WAII VERSION OF DASH PROGRAM

```
   T     D   G(T,D)
   1     0    1.399
   2     4    0.225
   2     6    0.049
   2     7    0.465
   2    10    0.541

   F  LEVEL=0     1     2     3     4     5     6     7     8     9    10
0.1308         0     2     0     0     0     2
0.1594         0     0     3     0     0     0     1
0.1856         0     0     2     1     0     1
0.0692         0     1     0     1     2
0.4550         0     1     0     2     0     1


>---
```

=$CHANGE A2=13, D2=12, PI2=.95, PKT3=.75$

```
...PHASE 1 IS UNCHANGED.

...PHASE 2

   4 ITERATIONS.  ELAPSED TIME:    14.00 SECONDS.

(T,A,D)=(   4,   13,   12) AB=11 MNT=    2 MXT=    4 PKT=0.86500 PI=0.95000
CODE=    111100101

TSWOB(UA)=     0.0077      TSR=     2.7777      TSFR=     2.7456

IS=     2.7250

TT=     1.2750

   T     D   G(T,D)
   1     0    0.005
   3    10    0.016
   4    12    0.986

   F  LEVEL=0     1     2     3     4     5     6     7     8     9    10
0.9001         0     1     0     0     3
0.0999         0     0     1     1     2
```

FIGURE 8—Continued
A SAMPLE SESSION WITH WAII VERSION OF DASH PROGRAM

```
...PHASE 3

   0 ITERATIONS.  ELAPSED TIME:   10.00 SECONDS.

(T,A,D)=(   4,   12,   12) AB=10 MNT=    2 MXT=    4 PKT=0.75000 PI=0.56901
CODE=    111100101

TSWOD(UA)=     0.0625      TSR=     1.2398      TSFR=     1.2398

TS=     1.4521

TD=     2.5479

   T     D    G(T,D)
   1     0     0.005
   3    10     0.016
   4    12     0.986

   F  LEVEL=0     1     2     3     4     5     6     7     8     9    10
0.1308        0     2     0     0     0     2
0.1594        0     0     3     0     0     0     1
0.1856        0     0     2     1     0     1
0.0692        0     1     0     1     2
0.4550        0     1     0     2     0     1


>---

=$CHANGE QUIT=1$

PROGRAM STOP AT 0
```

BIBLIOGRAPHY

Sources Cited in the Text

Reference Number

1. J. D. Matheson. Preferential Strategies with Imperfect Weapons (AR 67-1, Analytic Services Inc., April 1967, AD 813 915).

2. Final Report on Preferential Strategies for Urban Ballistic Missile Defense [Hq USAF (AFRDQ) and Analytic Services Inc., December 1968, AD 847 673].

3. J. D. Williams. The Compleat Strategyst (New York: McGraw-Hill, 1954).

4. G. Owen. Game Theory (Philadelphia: W. B. Saunders Co., 1968).

5. F. S. Hillier and G. J. Lieberman. Introduction to Operations Research (San Francisco: Holden-Day, 1968).

6. FORTRAN (TSS/645 Reference Manual, Rome Air Development Center, October 1968).

7. C. D. Hodgeman, editor. C.R.C. Standard Mathematical Tables, 12th edition (Cleveland: Chemical Rubber Publishing Co., 1962).

General Sources

A. R. Eckler. Mathematical Models of Target Coverage and Missile Allocation (Technical Memo MM68-4113-19, Bell Telephone Laboratories, 19 June 1968).

J. von Neumann and O. Morganstern. Theory of Games and Economic Behavior (Princeton: Princeton University Press, 1953).

UNCLASSIFIED

Security Classification

DOCUMENT CONTROL DATA - R & D

Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified

1 ORIGINATING ACTIVITY (Corporate author) Analytic Services Inc. (ANSER)	2a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED 2b. GROUP —

3 REPORT TITLE

OPTIMAL GROUP STRATEGIES

4 DESCRIPTIVE NOTES (Type of report and inclusive dates)

ANSER Report

5 AUTHOR(S) (First name, middle initial, last name)

Jay E. Israel

6 REPORT DATE February 1970	7a. TOTAL NO. OF PAGES 86	7b. NO. OF REFS 9
8a CONTRACT OR GRANT NO F44620-69-C-0014 b. PROJECT NO c. d.	9a. ORIGINATOR'S REPORT NUMBER(S) AR 70-1 9b. OTHER REPORT NOS (Any other numbers that may be assigned this report)	

10 DISTRIBUTION STATEMENT

Distribution of this document is unlimited. It may be released to the Clearinghouse, Department of Commerce, for sale to the general public.

11 SUPPLEMENTARY NOTES	12 SPONSORING MILITARY ACTIVITY Directorate of Operational Requirements and Development Plans Hq USAF

13 ABSTRACT

This ANSER Report generalizes the theory of precommit antimissile defense strategies by analyzing the case in which the defense has the ability to assign interceptors to defend groups of targets instead of individual targets only. The defense's objective is to choose the mix of group sizes and the number of interceptors assigned to each group in such a way that target survival is maximized in the event of attack. In this report, the analytic basis is established for a technique to study group defense strategies. Also, a computational procedure is presented for determining optimal attack and defense allocations numerically and for calculating the expected number of targets surviving. Game theory and linear programing are among the analytic techniques used. (U)(Author)

DD FORM 1 NOV 68 1473

UNCLASSIFIED

Security Classification

14 KEY WORDS		LINK A		LINK B		LINK C	
		ROLE	WT	ROLE	WT	ROLE	WT
12/01	Game Theory						
15/07	Military Strategy						
15/03.1	Antimissile Defense Systems						
12/01	Mathematical Analysis						
12/02	Operations Research						
12/01	Optimization						
12/01	Linear Programing						
15/07	Strategic Warfare						
15/03	Defense Systems						
09/02	Computer Programing						
12/01	Combinatorial Analysis						
17	Impact Prediction						
15/07	Interception Probabilities						

DISCLAIMER NOTICE

THIS DOCUMENT IS BEST QUALITY PRACTICABLE. THE COPY FURNISHED TO DTIC CONTAINED A SIGNIFICANT NUMBER OF PAGES WHICH DO NOT REPRODUCE LEGIBLY.